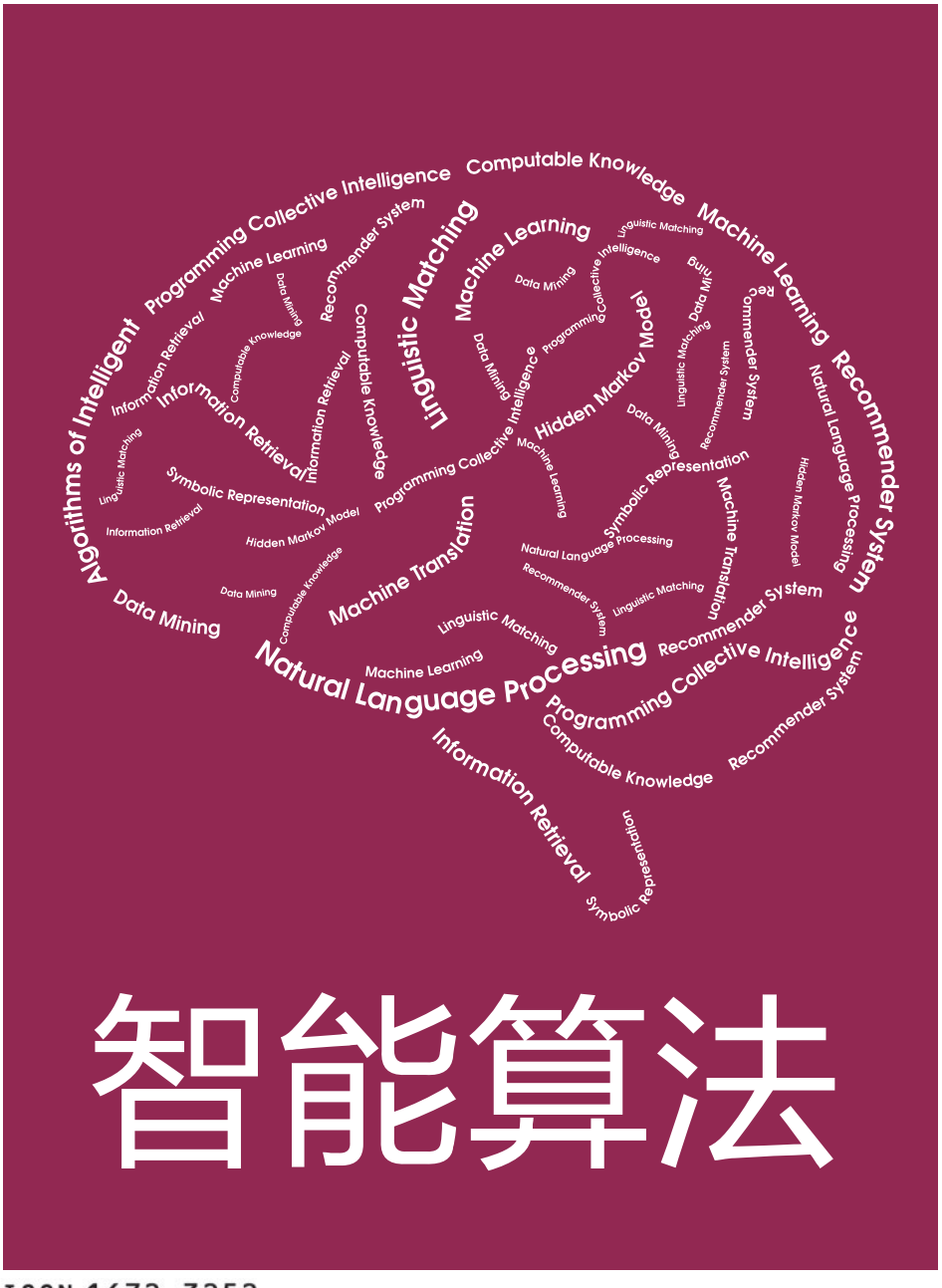
2012.07

PROGRAMMER 程序员



智能算法

28 云计算基础架构特别专题



没有最好，只有最合适
IBM 韩忠恒带你解读
《中国云计算基础架构建设指南》

圆桌论坛
云计算基础架构建设

16 创新与实践
第四届中国云计算大会观察

22 WWDC 2012：开发者为王

25 异构系统架构编程的现状和展望

70 轻敏捷
开放平台的制胜之道

74 如何设计高转化率的网站

81 Windows 8
能否推动移动互联网新变革?

98 公共IaaS服务的实测与选择

112 Swift
OpenStack对象存储

122 用 C++ 进行函数式编程

130 Mac OS X 文件系统的来龙去脉

134 “新”科学家
Stephen Wolfram

ISSN 1672-3252

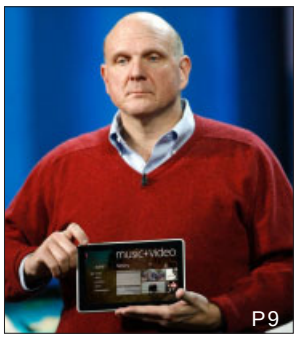
9 771672 325128 07>

邮发代号：2-665　定价：15元

www.programmer.com.cn　www.csdn.net

Contents





微软正式发布Windows 8发行预览版

资讯

5 外刊速递

8 新闻

10 新产品新工具

12 程序天下事

16 创新与实践

——第四届中国云计算大会观察

20 灵活、开放、自由的统计语言

——第五届中国R语言会议剪影

22 开发者为王

——苹果WWDC 2012见闻

25 异构系统架构编程的现状和展望

——AMD副总裁Leendert van Doorn专访



封面报道：智能算法

智能应用每时每刻都在影响着我们的工作和生活，然而对于许多软件开发人员来说，构建智能应用的技术——智能算法依然披着神秘的面纱。一方面，这些技术的潜在价值可以带来巨大的经济回报，商业公司往往不会轻易示人；另一方面，几乎所有的相关技术都源自学术研究，倘若缺乏引领，则难以深入其中。本期封面报道，我们将从搜索技术、社交网络数据挖掘、机器翻译、推荐引擎等角度，结合具体应用，为你揭开智能算法的面纱。

39 地图和本地搜索的最基本技术

42 搜索引擎的查询意图识别

46 社交网络：数据科学家眼中的金矿

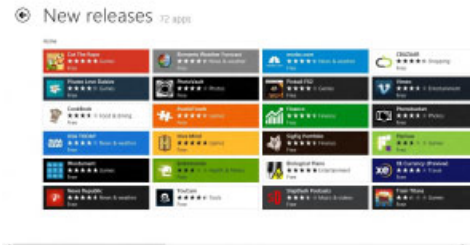
50 标签传播算法在微博用户兴趣图谱的应用

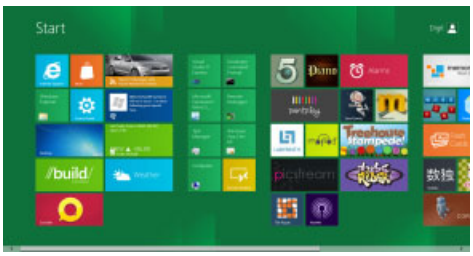
54 基于大规模语料的新词发现算法

58 内容推荐和优化的在线模型

62 “人工+智能”：机器翻译应用的未来

66 基于HMM的中文整句输入法实现





Windows 8能否推动移动互联网新变革？ P81



管理

70 轻敏捷
——开放平台的制胜之道
本文结合轻敏捷方法的三条原则，讲述敏捷团队所需要完成的工作、核心实践及所依赖的工具。。

74 如何设计高转化率的网站
本文分析了在流量既定的情况下，影响网站转化率的因素，同时指出了设计高转化率网站的要点。

76 听市场需求，而非个人需求

77 如何辅导新人

移动

80 做正确的加法

81 Windows 8能否推动移动互联网新变革？
本文从移动设备、Metro UI、编程的变化、Windows Store四个方面分析了Windows 8在移动互联网时代带来的影响与面对的挑战。

84 游戏画面的印象价值和审美属性分析

87 Cocos2D-X for XNA游戏开发指南（下）
——《TweeJump》项目实战

92 iOS即时语音聊天技术实践
本文讲解了如何在iOS设备上实现语音聊天应用开发，主要介绍了语音应用开发中的语音录制、播放、编解码等技术。

云计算

98 公共IaaS服务的实测与选择
本文基于多个公共IaaS服务的实际测试数据，从架构、存储等多个方面，阐释了影响IaaS服务性价比的因素，同时给出了中肯的建议。

103 腾讯分布式数据仓库解析

108 高可用的HDFS架构剖析
本文将对HDFS的NameNode内核的原理、数据结构及社区中最新的HA解决方案进行剖析，同时给出了社区的架构走向。

2012年07月刊　总第237期

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行总编：刘江　Executive Editor-in-chief：Liu Jiang
执行主编：孟迎霞　Managing Editor：Meng Yingxia
编辑部主任：董世晓　Director：Dong Shixiao
责任编辑：陈博　杨爽　卢鹄翔
Editors：Chen Bo　Yang Shuang　Lu Dongxiang
特邀编辑：蔡学镛　池建强　冯大辉　高博　肖新光　杨栋　余晟　周爱民
Contributing Editors：Cai Xueyong　Chi Jianqiang　Feng Dahui　Gao Bo
Xiao Xinguang　Yang Dong　Yu Sheng　Zhou Aimin
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：朱凯　Art Editor：Zhu Kai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
E-mail：editor@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
E-mail：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
E-mail：market@csdn.net

发行部
Distribution Dept.010-64351431
E-mail：sales@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Zhongrun Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

本刊文章版权所有 未经许可不得转载
发现装订错误或缺页，请将杂志寄回本刊读者服务部调换

112 Swift：OpenStack对象存储
OpenStack Object Storage（Swift）是OpenStack开源云计算项目的子项目之一，被称为对象存储，提供了强大的扩展性、冗余和持久性。本文将从架构、原理和实践等几方面讲述Swift。

技术

118 Facebook Folly代码分析
Folly是Facebook的一个开源C++11组件库，它提供了类似Boost库和STL的功能，用于满足大规模高性能的需求。

122 用C++进行函数式编程
《Quake》作者Carmack认为追求函数式编程有着实在的价值，但劝说所有程序员抛弃C++，转而启用边缘语言，是不负责任的。

百味

128 新书上架

130 Mac OS X文件系统的来龙去脉（上）

134 “新”科学家：Stephen Wolfram

企业专栏

15 风起亚洲公有云
——Joyent技术在中国的崛起

36 UC梁捷专栏：Web App的未来

96 中国首台云电脑全面解析
——天霆云计算董事长谈天霆专访

116 十年磨一剑，开源促创新
——英特尔开源技术总监Dirk Hohndel专访

133 MSUP三步曲：MPD之因团队施教
——聚焦软件研发中心的快速成长

云计算基础架构特别专题

28 云计算基础架构：没有最好，只有最合适
——IBM韩忠恒带你解读IDC《中国云计算基础架构建设指南》

30 云计算基础架构建设一席谈



Wired

2012.6

两轮交通工具的未来——陀螺仪稳定式两轮车

美国旧金山Lit Motors公司以改变传统摩托车弊端为己任。

C1型电动车是Lit Motors公司首款全电动、全封闭、二轮二座电动车原型。2014年将推出量产车，届时C1车型将配置安全气囊、安全带和可与智能电话连接的信息系统。不过，令人大开眼界的另有其他——乘客舱下方装有两个陀螺仪，C1因此能够始终保持直立，还能像最灵活的本田、雅马哈和杜卡迪摩托车一样急转弯。要用多大力气才能令C1侧翻在地呢？据Lit公司称，若要把C1撞翻在地，需一头小象从侧面进攻方可。

目前面世的C1原型车采用后轮驱动，不过，量产车将采用两轮驱动，通过轮毂电机供电，功率高达110马力；生产规格规定车重800~900磅；Lit公司估计从0英里每小时启动至60英里每小时需时6秒左右，最高时速为120英里/小时；装在底板上的8kWh锂离子电池每充电一次可以辆行驶220英里，充电时间与充电电压有关，美国标准家庭120V电源为C1充满电需时6个小时左右，而220V电源为C1充满电需时4个小时左右。

在见证者陪同下，Lit Motors创办人兼首席执行官Danny Kim试驾了C1原型车。结果，行驶最初一段路程后，软件出了岔子，只有一个陀螺仪能够正常工作。一位工程师拿出手提电脑，经过一连串系统测试和重设，大家再度前进。接着，皮带断了——试车过程中的这些故障倒让Kim和他的团队挺高兴，因为这样就可以让见证者看到，在Lit团队将陀螺磁体从底盘中取出后，C1车靠硬件做到了让电动车直立不倒……见证者认为，C1的运转的确不出广告所言，不过很明显，若要迎来黄金时代，还有很多事情要做。

首款C1原型是Kim和他强大的六人团队的作品，耗资20万美元，后续测试、升级、改进、规范化还需要数百万资金，而让C1成为美国各州的合法产品更需要大型汽车制造厂伸出援助之手。

播下种子就有希望，两年内，你我有望开上C1电动车。

Facebook称霸网络，克星何在？

自诞生至当下，Facebook已存世8年，独霸7.5亿多用户。人们花在这个网站上的平均时间从2009年的每月4.5个小时延长至当下的每月将近7个小时，是竞争对手的两倍还多。

Facebook霸气渐显，表明这个平台有可能充分代表了网络时代的第三次重大演变。首先，在互联网的推动下，点对点架构以及分组交换数据的重要组织原理得以盛行；接着，网络迎来了一系列新的主流隐喻，这些隐喻从根本上看都具有文学本质，都是由“页”和注脚似的链接组成的网络。尽管这些隐喻功能十分强大，但都以数据为核心，而不是以人为核心。在计算机专家看来，这似乎并不是缺点。可是，大部分普通人却不会自然而然地通过域或链接来组织我们的世界，相反，普通人心里的世界是通过由朋友、家庭和同事构成的社交网络来反映的。

Facebook所向披靡，这也许正是在它进行资本史上最成功募股之际批评之声渐响的原因。在有其他选择时，容忍问题公司并非难事，换一家就行了；可当一家独大时，每一个小小的过失都会被放大。几年前，关于Facebook的批评不过是大大浪费了我们的时间，如今，抱怨来得猛得多：据说，Facebook对我们的核心社会价值、对隐私、对网络本身都有威胁。

Facebook正确地指出自己面临众多竞争对手，例如，Google+不到一年就积累了1.7亿用户； Foursquare、Path、Spotify等小型网络也达到突破性规模。不过，这些网站与Facebook比起来可谓小巫见大巫。Facebook胜者为王、横扫网络、势如破竹——加入一个网络的用户越多，网络就越有用，就能吸引更多人加入。1990年代的Windows平台和2000年代的Google商务广告都曾发生过规模相当的“逃跑效应”，Microsoft和Google最终招致反托拉斯调查。从Facebook的S-1文件看来，Facebook似乎更担心竞争暗箭，而非垄断质疑。可历史告诉大家，反托拉斯调查有可能成为最大的威胁。如果Facebook继续从网站向媒体变身，扎克伯格和他的团队所拥有的Facebook恐怕必须走向开放式标准，或至少其中一部分将走向开放式标准。

想象力引擎：下一代视频游戏撼动玩家

在美术设计、撰稿和情节之外，每一款视

频游戏都拥有一套精心设计的软件，称为游戏引擎。这是一套程序和算法的集合，是一张元素周期表，凭借它，游戏编程人员和设计人员可以创造出游戏玩家期望见识的丰富多彩的游戏世界。照明、背景、人工智能，这些都是游戏引擎的功劳。游戏工作室将这些元素组合在一起后，游戏引擎还要再干一份工作——在游戏启动时运行这些元素，同时对没完没了的交互、情节和结局予以控制。

有些开发者自己动手开发引擎，但绝大部分开发者依赖于其他公司的创造。使用Epic公司UE（Unreal Engine）游戏引擎的开发者成百上千。该系列游戏引擎诞生于1998年，是Epic Games公司创始人Tim Sweeney及其团队的开发成果。经历过三次大变迁的UE，早已成为游戏行业的奠基石。现在，最新一代UE引擎——UE4也已经万事具备，即将面世。

UE4完全可以代表今后十年的游戏基础，恐怕Microsoft和Sony都得重新考虑如何提高新硬件的马力，以求与引擎配合。UE4将使游戏开发如行云流水般顺畅——游戏开发工作室目前用两年完成的工作，未来将只用12个月即可完成。最重要的是，它将让过去十年一统江湖的游戏情境形同木偶表演。使用以往的游戏引擎时，只要飞起一粒余烬，就足以让游戏速度大大下降，更绝无可能展现余烬阵阵飞扬的效果。使用UE4却可以在硬件性能支持的情况下呈现数百万这样的粒子。2012年3月，Epic公司在游戏开发者大会上演示了这款游戏，长达153秒的短片向微软、索尼等来宾演示了多年来让他们捉襟见肘的各种特效：镜头光晕、熔岩流、环境破坏、火灾和数英里以外的细腻景观。另外，倘若演示短片中的骑士们算得上真人的话，图像的逼真性也令人叫绝。

UE4的应用范围十分广泛，虚拟现实、医学仿真乃至电视及电影制片，似乎无所不能。不过，UE4根本上还是一款视频游戏引擎。6月，游戏界将迎接它的到来。



Communications of the ACM

2012.6

颠覆游戏的人

Judea Pearl对概率与因果关系重要性的研究将助力人工智能革命。

2011年图灵奖得主Judea Pear一度坚定地认为，人类是通过思维中的概率引擎进行推理，可是由于将标准概率分布应用于有关变量的数目会使事情复杂程度呈指数级增长，因此概率并未受到重用。心理学理论中提到过儿童识字方法，受此启发，Pearl开发了几个贝叶斯网络，通过这些图形模型表明信念如何对新的观察结果作出反应并进行传播。他所开发的算法综合了图形理论和概率论，系统十分简单，足以让计算机通过概率方法进行推理。

可Pearl很快认定，人类思维中并没有什么“概率引擎”，相反，人类根据对原因和结果的理解进行推理，从而能够问出“要是……会怎么样？”这样的问题。如果执行A事件，结果应该为B，分析者就能够考虑出现这种结果的可能性。“我们的思维中有一个因果引擎，我们以概率为其辅助，”Pearl说：“科学家正是这样积累科学知识的。是什么导致了什么？”同样，因果推理还让分析者有能力对反设事实进行推理。如果本应该发生的A没有发生，那么就能够询问原因。如果吞下一片阿司匹林却犯了胃病，那么就可以作出不该吃阿司匹林的决定。

哲学家们对因果关系的思索已有数百年，对于这一套，从医生到律师都有所顾虑。可是Pearl想出了办法用数学方式对此进行表达，探讨内容得以改变。Pearl的伙伴认为因果关系领域因他得以复兴，事物观察方法有了某种新内容。Pearl的工作不仅对计算机科学有意义，对于经济学、传染病学、疾病诊断都有意义，尤其对于机器学习和自然语言处理领域意义重大。

伯尔敦促计算机科学家，在别人说某台机器无法效仿人类时不要畏缩，“当你看见人类的某种现象时，计算机肯定能够模仿”。

计算机科学家应该关注网络冲突及美国国家安全局政策，为什么？

今年，网络安全引起媒体热议，反映出世界各地公共政策对这个问题的关注。例如，伦敦《星期日泰晤士报》援引以色列首相本杰明·内塔胡亚尼的言论：“以色列必须转型为全球网络强国”；又如，中国政府通过国家计算机网络应急协调中心发布报告，指出中国在2010年遭遇过将近50万次网络攻击，其中约有一半来自海外。有人认为不可能将网络攻击归罪于某个特定国家，也有人认为国家战略所假设的国家赞助是发生此类攻击的必要条件。不过这两种说法显然都过于简单。

政策制定者拥有应对各种危机的经验，但他们对网络世界的理解却有些例外，显得笼统而不全面。尽管如此，当出现网络危机时，不管所拥有的信息和知识是否合适，政策制定者还是要据此做出决策。计算机科学家的身份令他们扮演着帮助规划国家和国际网络冲突政策的角色。如今正处于事关网络空间战争与和平的政策论战的初期阶段，专业计算机科学家的观点应当得到尊重。无论具体观点如何，在计算机科学社区对这个问题进行对话和讨论将有助于政策制定者对这一领域做出更有道理的决策。

计算机科学家需要关注这样一些问题：攻击评估；计算机地理位置定位；限制网络攻击范围的技术；采用何种方式侵入对方的计算机系统，才能让对方看出这是利用式侵入，而非攻击性侵入？在犯罪式网络攻击和黑客式网络攻击频频发生、令人不堪其扰的背景下，两个达成“网络停火”协议的国家如何知道对方是否正在遵守协议中的各项条款？如何尽量避免两个国家之间出现催化式网络冲突（催化式网络冲突是指在第三方引发下使得另两方发生冲突——有可能第三方模拟另两方中的某一方）？以及如何防止政治/军事对手在网络空间中的小冲突演变成大冲突？

探究传言在社交网络上扩散如风的原因

在社交网络上，消息的传播速度迅疾如风，令人叹服。社交网络的结构以及消息传播的过程并非为实现这种效果而设计，相反，根本没有人对此进行过任何设计，这种效果不过是随机的、离散的演变结果。

社交网络方便了信息（传言）的传播，这种假设是否正确？如果正确的话，是哪种属性使得这种结果成为可能？为了回答这个问题，研究者通过几张图（如下所列）对一个简单的传言传播过程进行仿真，这几张图反映了现有的几个大型社交网络的结构。例如，传言从Twitter网络上的一个随机节点开始传出，仅经过8轮传播，就到达4560万网络成员，而总成员才5120万。

研究者所用图形：

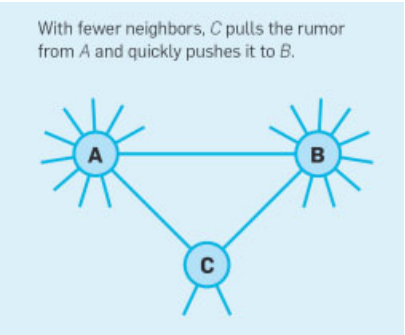


图1 传言如何从大度数节点A传送至B：由于相邻节点数目很多，A和B需要更多时间来推或拉传言

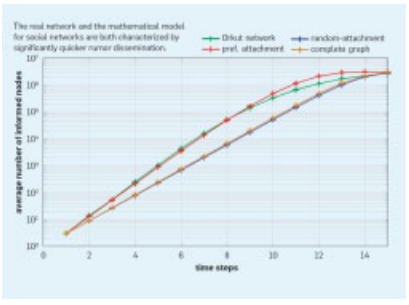


图2 在Orkut网络、优先连接图、随机连接图以及完全图中，收到信息的节点的平均数目是时间的函数，图形大小： 节点数n= 3,072,441，密度参数m=38

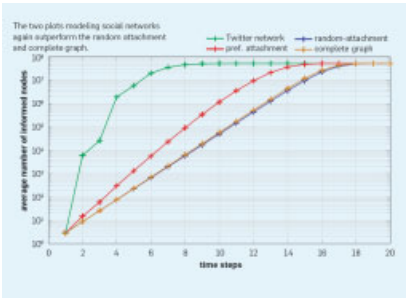


图3 一段时间内收到信息的节点的平均数目，研究对象为Twitter网，比较对象为优先连接图、随机连接图以及完全图，图形大小： 节点数n=51,217,936，密度参数m=32

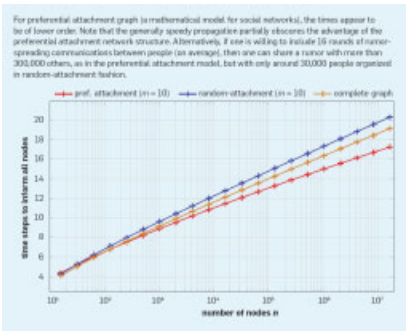


图4 大小不一的各种网络中的所有节点得到信息所需要的平均时间，数据显示出随机连接图和完全图中存在对数相关性

研究者还通过各种社交网络的一个抽象模型对传播过程进行分析，该抽象模型称为PA图，由Barabasi与Albert在1999年提出。在Doerr等人的文章中，研究者设计了一个数学证明，指出这些社交网络中的传言速度要比采用其他拓扑结构的网络快得多，甚至比具有节点与节点通信链接的网络（完全图）都要快。研究者的观察结果可以对此作出解释：相邻节点相对较少的节点在具有大度数的节点（Hub）之间构成捷径，这些大度数节点由于可能发生交流的伙伴数量极大，因此相互之间的直接对话往往更少。具有众多连接的Hub和只有少数朋友的一般用户之间的有效互动至关重要。

（感谢译者李芳支持）

诺基亚裁员

6月14日，诺基亚宣布，将在2013年底前在全球裁员1万人，并计划关闭位于德国和加拿大的一些部门，以及关闭位于芬兰的生产设施，诺基亚中国区此次也将受到波及。自从艾洛普出任CEO以来，诺基亚已经裁员超过4万人。目前诺基亚在全球的员工总数约为5.35万，此次裁员，几乎占到员工总数的五分之一。在裁员的同时，管理层也进行了大幅调整，首席营销官Jerri DeVard、手机部门负责人Mary McDowell、市场部负责人Niklas Savander三位高管将从公司离职。

诺基亚还宣布，已将独立运营的奢侈品手机业务Vertu出售给私募股权投资公司EQT，该公司还将评估其他非核心资产的前景。

另有消息传出，诺基亚已停止Meltemi系统开发，Qt部门也面临着解散，而基于此的MeeGo系统看来也将终结。Qt for Symbian&Meego的成员Eero Penttinen在Twitter上宣布了离职的消息，他表示，诺基亚的下一个战略将基于Java。

对于这次裁员，Linus Torvalds评论说，诺基亚选择Windows Phone犯了一个错误，但他理解他们为什么这样做，因为诺基亚的新CEO是前微软员工。

苹果抛弃Google地图

WWDC落幕，对于iOS 6移动操作系统来说，最大的变化就是将Google地图从首屏拿掉，苹果第一次将iOS设备中的地图应用完全把握在了自己手中。

苹果最近几年一直没有停止过对地图技术公司的收购，Placebase、Poly9和C3 Technology都被苹果收入囊中，其中C3 Technology为这次iOS 6中的3D飞行模式提供了核心技术。这些收购动作都不是小手笔，要知道，诺基亚2007年收购导航软件商Navteq的价格就高达81亿美元。可以想见，今天的这步棋已是布局多年的结果。



尽管已拥有相当的技术储备，但胜负依然难料，Gartner分析师Brian Blau在接受《连线》杂志采访时说：“地图水很深。它的技术天生复杂，数据集非常庞大，保证导航的准确性非常困难，此外还要考虑交通数据和兴趣点——尤其是目前的GPS定位还不是很准。”

技术背后，更被人看重的是，这些行动意味着苹果不但发出了与Google的分手宣言，更被认为是向Google发出了开战信号。不仅针对Android，苹果还试图对iOS、网页和移动设备“去Google搜索化”，瓦解Google赖以生存的基础。

创业冬天



Paul Graham向其投资伙伴发出邮件，表达了他对Facebook IPO的不佳表现将打击初创公司融资市场的担忧。他担心未来的创业者或许将难以融到一分钱，不管估值是多少。此外，之前对于曾以很高估值融资的企业也将面临“流血融资”，这将对企业造成很大伤害。

“尤其是那些以优越的条件融得很多资金，并大手大脚地花钱，同时不注重盈利的公司”，Paul Graham认为这种创业公司将在市场收紧时被摧毁。

不过针对Paul Graham的观点，也有人持不同意见。比如曾经投资过Twitter、Tumblr、Foursquare、Zynga等公司的纽约投资人Fred Wilson就不以为然，按照他的计算，Facebook目前市值减现金后价值400多亿美元，只是年收入的10倍左右，很值。而《Business Insider》也认为，Facebook上市有10亿美元流入其员工和投资人腰包，为硅谷带来了充沛资金。

“开源不是‘让我们围着篝火唱歌，使这个世界变得更美好’。只有人人因为自私的理由贡献时，开源才能真正创造奇迹。”

——Linus Torvalds在接受BBC采访时如是说。不过他所指的“自私”并不是经济回报，而是追求乐趣。

“在推出全新的产品之时，库克则将面临最大的挑战，或许那时业界对他的评议标准会迥然不同。”

——市场研究机构Forrester Research分析师Charles Golvin认为，即使人们不满意苹果公司对现有产品的更新，库克也不会受到责难，而真正的挑战将出现在今年底或明年初。

“Bitcoin像BitTorrent网络协议一样坚挺，不会轻易被打败。监控部门应该放手，Bitcoin有成为人们生活好帮手的潜质。”

——经历了初期的“集万千宠爱”到如今“云淡风轻”的华丽转身，Bitcoin软件首席设计师Gavin Andresen和George Mason大学研究员Jerry Brito都认为各国的政府不会完全关闭Bitcoin系统。

“我很高兴成为第一个使用SGI UV 2的人，这台电脑将有助于我们更好地探索宇宙。”

——著名天体物理学家斯蒂芬·霍金表示，在新系统的帮助下，英国研究人员可以更快地掌握最前沿的研究方法。SGI表示，超级计算机可以让研究人员免于陷入细节，更有利于取得成果。

“现在公司完成了IPO，推出了一些新的东西，对我来说就是最好的时机。”

——Facebook CTO Bret Taylor离职创业，他称自己在Facebook的这段时间是“职业生涯中实现个人追求的岁月”，而自己的离职也只是硅谷人生活最为平常的一部分。

“作为Web设计师，我们在考虑社交媒体时，仅停留在每个页面底部都加上一个‘分享’按钮，而很少有超越。”

——英国网页设计公司Headscape创始人Paul Boag指出，社交媒体应该是用户体验的重要部分，而很多网站将其作为附属物被随意添加到某一个网页中。

“对这种黑客攻击行为所作出的回应不该只是更改你的LinkedIn密码，而是更改任何使用相同密码的网站。”

——包括美国职业社交网站LinkedIn、CBS旗下音乐网站Last.fm和在线约会网站eHarmony在内，几家互联网公司都发生了上百万的用户密码被盗事件，安全专家呼吁用户增强安全意识。

CSDN十大资讯

2012年5月26日–6月26日

01 蒂姆·库克讽刺Windows 8不伦不类

作者 / 汤姆

D10数字大会上，蒂姆·库克对微软的Windows 8提出了尖锐批评。库克说：“在我看来，平板电脑和PC机是完全不同的产品，许多厂商愿意推出一款介于两者之间的混合型电脑，但苹果不会这么做。因为混合型产品无法在PC和平板间取得平衡，反而会影响双方优势的发挥。”

02 同学们，一起来可视化编程吧

作者 / Kramer

Hultman Kramer是洛杉矶电脑公司Oblong Industries的CEO，这家公司曾为电影《少数派报告》开发手势界面。Kramer指出，可视化编程环境十几年前就已经出现，并不是这两年才兴起的概念，但这种方式之前没有流行起来，其中有很多因素值得我们思考。

03 微软正式发布Windows 8发行预览版

作者 / 魏兵

微软6月1日推出了Windows 8发行预览版，这将是最后一个Windows 8预发布版本。微软官方在博客中透露，如果有关Windows 8和Windows RT的反馈和测试结果符合预期，新版操作系统将在接下来的两个月进入RTM阶段，这也将预示着微软能够在年底正式推出Windows 8和Windows RT。

04 IE 10或成在线广告噩梦

作者 / Ryan Singel

微软宣布最新版的IE 10包含“请勿追踪”功能，该功能并不意味着阻止Cookie，而是给每个网站发送信息，告诉这些网站浏览者不希望被追踪。这在主流浏览器中尚属首例，对广告巨头造成了潜在威胁，这其中就包括微软的竞争对手Google。

05 2012苹果全球开发者大会开幕

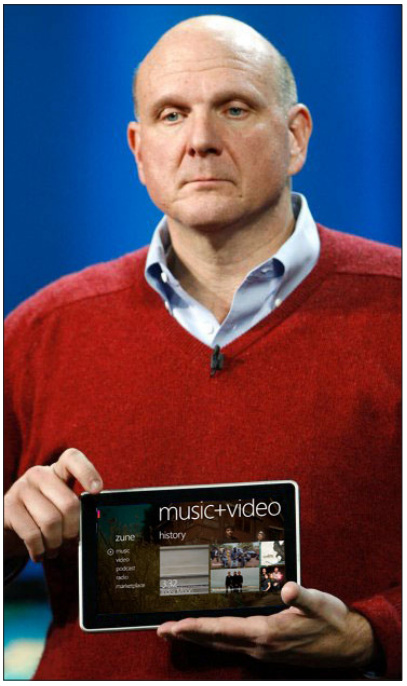
作者 / 魏兵

2012苹果全球开发者大会于6月11日–15日在旧金山召开。在主题演讲中，苹果公布了OS X、iOS的最新进展，并宣布了新的笔记本产品。本届大会包括100多场演讲活动，涵盖应用服务、开发者工具、图形、媒体、游戏、Safari和Web等多方面内容。

06 Linus Torvalds获2012年千禧年科技奖

作者 / 张祺

千禧年科技奖是芬兰科技学会颁发的一个奖项，自2004年起每两年评选一次，目的是奖励在科学技术和IT界中有重大贡献的人物，被誉为“工程界的诺贝尔奖”。2012年该奖项的获得者是Linus Torvalds和日本京都大学教授山中伸弥。



07 为什么C语言屹立不倒？

作者 / Caleb Garling

Leo Meyerovich和Ari Rabkin希望通过自己的研究，解开编程语言世界的丛林法则。他们在探寻一个问题——为何C语言虽垂垂老矣却能屹立不倒？在研究中他们发现，除却新兴语言本身存在的问题，这里面还有一个要素——许多程序员觉得学习新语言过于辛苦和困难。

08 跨平台语言Haxe

作者 / 杨鹏飞

程序员有两种，一种为了解决问题使用不同的语言，另一种为了解决问题发明新的语言。Haxe就是这样一门新兴的开源编程语言，开发者称：“与Java、C#等语言相比，Haxe是真正的多平台语言。不论开发原生iOS、Android程序，还是Web应用，Haxe都可以胜任。”

09 图解C++、CoffeeScript、Ruby复杂度

作者 / 王然

大多数人都同意C++是门相当复杂的语言，不过它究竟有多复杂？CppRocks网站对C++、CoffeeScript、Ruby这几门语言的必学功能的复杂程度进行了比较。结果显示，C++的语言特性总共涉及18个范畴、186个概念，几乎是Ruby的2倍、CoffeeScript的3倍。

10 顶级域名圈地，Google申请最多

作者 / 张勇

根据ICANN最新公布的一份顶级域名申请名单显示，包括Google、Amazon、微软等公司在内，都申请了多个顶级域名，其中Google申请的顶级域名数量最多，达101个。在这些域名中，不仅仅有与公司业务联系紧密的.youtube，也还有奇特的.dog域名。

2012年及未来10大移动技术

IT研究与顾问咨询公司Gartner日前针对IT企业和用户发布了2012年及未来移动技术预测，认为以下10项移动技术应当明确位列于各企业的移动战略之中。

01 HTML5

HTML5的重要性毋庸置疑，但它也还需要数年才能逐步进入低端设备市场。另外由于早期设备对HTML5支持并不完善，更新换代也需要一定时间。

02 NFC

虽然经常被认为仅是一项支付技术，但NFC其实是一种“借由触控来启动”的技术，日后可望带动多种应用程序的发展，例如智能海报、优惠券、移动门票，以及定位服务的验证和登记。

03 跨平台应用开发工具

在大多数地区，移动市场通常被3~5个智能平台占领。Gartner认为，在这种情况下，跨平台开发工具将扮演举足轻重的角色，它们还使应用程序延伸到手机和平板之外的设备，如智能电视。

04 定位技术

定位技术是实现高度个性化应用程序的第一步。目前基于移动网络、GPS、蓝牙、WiFi的定位技术已经相当成熟。但未来会有更多样化的需求，定位技术也需要进一步发展。

05 蓝牙4.0

蓝牙4.0标准定义了超低的峰值、平均和待机模式功耗，设备使用一颗纽扣电池甚至可连续工作数年之久。高端电子设备已开始使用这一技术，但真正普及还需数年。

06 802.11ac无线技术

IEEE 802.11ac是一个正在发展中的802.11无线计算机网络通信标准，它采用并扩展了源自802.11n的空中接口概念。理论上，它能够提供1Gb带宽进行多站式无线局域网通信。

07 M2M技术与智能产品

M2M是机器对机器通信的简称，它的潜在市场不仅限于通信，还将延伸到不同的应用方案，如安全监测、自动售货机、货物跟踪等。

08 增强现实

增强现实技术最早于20世纪90年代被提出，目标是在屏幕上与现实世界进行互动。随着随身电子产品运算能力的提升，增强现实的应用将会越来越广。

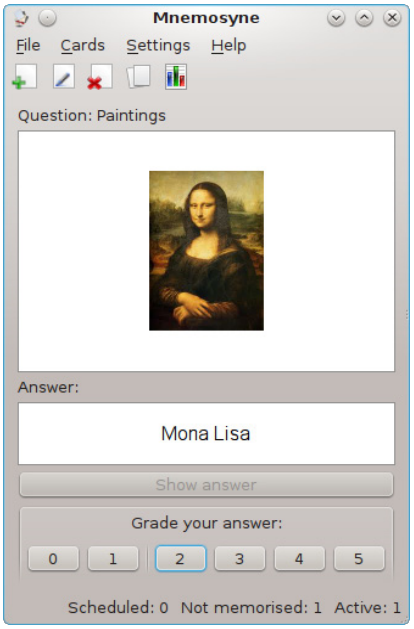
09 多平台移动设备管理

调查报告显示，企业软件在2012年至少需要支持3种智能设备平台，未来这一情况可能会变得更加复杂。企业软件的多平台移动设备管理能力是解决这一问题的关键。

10 LTE移动通信技术

3G技术是指同一无线网络提供语音和数据通信，但4G时代则变为全数据网络。LTE与WiMAX都被视为4G技术，能够提供100Mbps的下载速率与50Mbps的上传速率。

记忆辅助工具Mnemosyne 2.0



Mnemosyne以希腊神话中记忆女神命名，是一款采用间隔重复原理的记忆辅助开源软件，从2003年开发至今。采用类似原理的软件有SuperMemo和Anki。Mnemosyne使用的是SuperMemo算法的一个早期版本，并进行了适当修正。Mnemosyne可以用来帮助使用者学习外语、准备考试，以及所有使用者希望记忆在头脑中的信息，它也会收集使用者的数据，这些数据被用来进行长期记忆的研究。

经过4年的开发，Mnemosyne 2.0版问世，在新版中软件界面被进行了重新设计，便于开发者在其上开发插件，使用者可以更灵活地设置倒序等特殊卡片，新增了卡片浏览器和图形化的统计界面。此外支持同步协议，便于和移动端的软件同步。

GNU Emacs 24.1发布

科幻作家尼尔·斯蒂芬森曾经说：“Emacs的光芒胜过所有其他的编辑软件。它不只是更大更明亮，而是像正午的骄阳，让天际中所有的星辰都黯然失色。”

不过新出现的编辑器往往能抛开历史负担，更充分地利用操作系统提供的功能，为开发者带来创新性的体验，例如Sublime Text。老牌编辑器在这方面则往往显现出渐进式的发展，逐渐吸收已经被证实有效的特性。

GNU Emacs 24.1的新特性主要包括：加入了新的包管理系统，支持阿拉伯语从右向左输入文字，Emacs Lisp支持词法作用域，改进了色彩主题系统。

Google发布 AngularJS 1.0

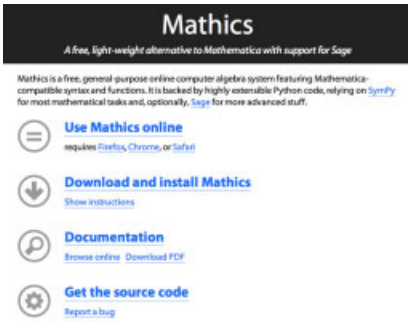
AngularJS是一款开源JavaScript框架，目的是增强浏览器端应用程序的MVC能力，减少Web应用程序对JavaScript的依赖。这种类型的应用也被称为单页面应用（Single-Page Applications）。

Google表示，AngularJS可以让开发者扩展HTML语法，以便清晰、简洁地表示应用程序中的组件，并允许将标准的HTML作为模板语言。AngularJS可以通过双向数据绑定自动从拥有JavaScript对象模型的UI视图中同步数据。

免费数学工具软件Mathics

许多领域的科研和工程研究都离不开数学计算，Mathics是一款开源通用计算软件，它支持使用与Mathematica兼容的语法和函数。Mathics使用高度可扩展的Python代码编写，依靠SymPy进行绝大多数的计算，并且可以通过Sage完成更复杂的计算任务。

用户除了可以在本地安装运行，还可以直接通过浏览器在线获得运算结果。开发者希望Mathics能在某种程度上代替Wolfram Research的商业软件Mathematica。



PyPy 1.9发布

PyPy是用Python实现的Python解释器，同时也是一款即时编译器（Just-in-Time Compiler）。它的目标是快速、高效，以及与CPython完全兼容。PyPy的优势主要在于：得益于即时编译器，程序在PyPy中的运行速度通常领先于CPython；相对于CPython，程序对内存的占用也更小；完全与现有的Python代码兼容；支持沙箱，可以在安全的环境中运行有风险的代码；此外，PyPy默认支持stackless模式，拥有更好的并行性能。

最新的PyPy 1.9中修复了大量Bug，同时也带来了性能的进一步提升，该版本与CPython 2.7完全兼容。

DNSPoD宣布DNS Engine开源

DNSPod是国内最早提供免费智能DNS产品的网站，日前宣布将其DNS服务器软件高性能递归解析服务器（dnspod-sr）开源。dnspod-sr是一款运行在Linux平台上的高性能递归DNS服务器软件，具备高性能、高负载、易扩展的优势。

通过架设dnspod-sr集群，可以替换运营商目前基于BIND的陈旧方案，减少运营成本；还可以用于公司、学校、政府等组织内部DNS，解析外部不可见的私有域名，提高上网速度。

网站测试辅助工具xip.io

如果开发者有许多终端设备需要测试，往往会选择在本地网络中进行。然而配置这样的本地网络却是一件痛苦的事情。在这种情况下，xip.io可以帮上大忙。

从本质上看，xip.io是一个可以定制的DNS服务，由大名鼎鼎的37signals开发。它能够将类似site.10.0.0.1.xip.io这样的域名解析成10.0.0.1。开发者在调试Web应用时，可以很方便地利用xip.io这一特性。比如开发者希望在iPhone上测试自己的Web产品，但以往通过输入IP地址只能访问一个应用实例，利用xip.io提供的域名翻译功能，则能轻松处理。



支持多点触摸，X.Org 7.7 发布

X Window System的开源实现X.Org发布了7.7版。根据该版本主要包含以下新增特性：触摸板及触摸屏现在支持多点触摸事件；Xinput扩展添加了一些额外的特性，如允许客户端跟踪来自输入设备的原始事件、滚动事件可让客户端执行平滑的滚动等；X.Org文档已被转换成DocBook XML格式，支持文档间的交叉链接；在Synchronization扩展3.1中引入了Fence对象。

此外，X Fixes扩展5.0中添加了Pointer Barriers，XCB库开始针对GLX及XKB扩展添加支持，不过目前尚未完成，显示及输入驱动也得到增强。按X.Org的计划，下个版本将为X.Org 7.8，预计将在2013年推出。



Mozilla Thimble

Mozilla推出的网页开发工具Thimble旨在帮助普通用户在线学习编写HTML和CSS，轻松创建和分享自己的网页和其他项目。

Thimble简化了Web页面的制作，因为它拥有一个直观的可视化编辑器，可以让使用者在浏览器中编写、修改、预览和调试HTML代码。制作完成后，单击即可发布和共享全功能的Web页面。Thimble还包含了一系列启动项目模板，以帮助用户快速入门。

命令行Shell工具新fishfish



fishfish是命令行脚本工具fish Shell的Fork版本，与原版相比，fishfish为用户带来了基于历史及补全、原生256色支持、Man手册补全、易用的Web配置、极快的运行速度等特性。

fish这个名字是friendly interactive shell的缩写，fish专注于交互应用、易用性和友好的用户体验。fish的设计目的是通过容易发现、记住和使用的方式为用户提供丰富、强大的功能 。

SpaceX引领太空探索商业化

根据SpaceX的统计数据，NASA在完成类似发射任务过程中，消耗的资金往往是SpaceX的4~10倍。

太空探索技术公司（SpaceX）的第一架宇宙飞船Dragon号，在Falcon 9运载火箭的搭载下，于5月22日顺利发射升空并进入了预定轨道。在环绕轨道过程中，飞船进行了一系列复杂的机动性和系统级测试，以保证在与国际空间站对接的过程中万无一失。5月25日，Dragon号达到国际空间站所在的位置，被空间站的机械臂捕获并完成了对接操作。在完成国际空间站物资补给以及运载太空站返回货物的任务后，Dragon号飞船于5月31日顺利返回地球，着陆在太平洋海域。

随着Dragon号飞船一系列预订计划的成功实施，SpaceX公司创造了多项航天领域的新纪录。Dragon号是第一架由私人商业公司设计的可载人宇宙飞船，在容积上能够搭载7名成员或相等重量的货物到达近地轨道，可以承担为国际太空站进行物资补给或人员轮替的任务。在完成此次发射任务之后，SpaceX公司率先成为有能力将太空舱发射进入地球轨道，并与国际空间站对接的商业太空发射公司。

对于SpaceX来说，这次发射与对接任务的顺利实施远不仅是那些具有开创性的历史意义，更重要的是，发射任务的顺利实施将为SpaceX带来后续数十亿美元来自NASA及其他机构的订单，用来为国际空间站输送人员和物资，并承接太空微重力环境对生物、材料、流体物理学等相关研究对象影响的各项实验任务。NASA认为将低轨道载人火箭发射及运载计划承包给商业公司，将比自行完成这项工作更有效率，同时NASA也可以将其资源和注意力集中在探索更遥远的外层空间上。

实事的确如此，根据SpaceX的统计数据来看，原先NASA在完成类似发射任务过程中，消耗的资金往往是SpaceX各项花销的4~10倍。此外，创始人Elon Musk还在SpaceX官方网站公布，公司通过精细化管理以及设计生产过程中的高度垂直集成，使得目前Falcon 9号火箭的标准发射费用仅为5400万美元，并将使用铝锂合金制造的猎鹰系列运载火箭做到部分重复使用，通过出色的成本控制下进一步下调火箭发射的价格成本。

美国航天工业具有深厚的技术积累和完善的产业链，此外还有数量众多且经验丰富的工程人员。有趣的是，在航天界之外不少IT企业和技术人员，也把关注点转向太空资源的开发和利用。与PayPal创始人Elon Musk创建SpaceX相似，DOOM、QUAKE、CS等游戏引擎的传奇程序员John Carmack已不满足自己在游戏业中取得的成绩，也创建了面向太空技术研发的Armadillo Aerospace公司，专注于研制能够垂直起降并且可以重用的运载火箭技术，为实现载人太空旅行而努力。Carmack主导的Armadillo公司还创建了X-Prize Cup航空航天展和Lunar Lander Challenge登月挑战赛，为航空航天爱好者提供了施展的平台和空间。此外Google赞助发起的Google Lunar X PRIZE，也为青少年了解航天领域知识和动手操作提供了支持。

创建于2002年的太空运输公司SpaceX，仅用了8年时间就顺利完成了Falcon 1号和Falcon 9号运载火箭的设计和制造，并同步研发了Dragon系列宇宙飞船，成为继美国、俄罗斯、日本、欧空局之后第五个与国际空间站对接的航天组织。或许在不远的将来，在SpaceX、Armadillo等商业公司的努力下，航天发射的费用能够降到目前发射费用的1/10，那么，距离普通人去外太空旅行的愿望应该也会越来越近。Carmack曾设立了奖金为一千万美元的X PRIZE大奖，用于奖励能够将三名乘客送到外太空的航天发射项目。在不远的将来，哪个团队将摘走X PRIZE大奖，让我们拭目以待。



高昂
中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

财务报表数据管理不能走“快餐”路线

企业信息管理创新≠用了什么新技术、新开发语言，关键在于创新怎样更好服务业务发展的问题。

又到了众多欧美企业财年的当口，甲骨文和埃森哲一起“应景”发布的一份报告向企业的财务主管们提了一个醒儿——“如果继续以现有的投资方式支持财务报表系统恐怕是在打水漂。”

这份报告是在对12个国家和地区调研的基础上提出的，通过对众多企业高层的访谈发现，近些年多数先进的企业持续在财务报表系统上进行投资，并试图将电子化财务管理与企业各方面的运营系统紧密结合，力图确保资金流、业务流的同步，但最终的结果不太理想：

- 68%的报表缺乏可见性；
- 84%的受访者表示财务数据以及其他支持系统的数据质量难以驾驭；
- 尽管过去三年里企业在财务报表系统中追加投入的投资比例不小，但还是有60%的系统很难刻画企业总体成本情况；
- 此外，很大一部分企业遭遇了因为财务报表系统各方面问题（比如处理效率），难以及时获得财务报告的情况。

报告虽然不长，但其中却点出了出现这种情况的主要原因——“快餐消费”方式下的分散建设，即系统的建设往往是一事一议，最后导致数据难以有效形成合力、真正为企业高层及时提供企业运营整体成本信息，财务数据的有效性影响了商业决策的有效性。当然导致这个问题的原因是多方面的，最主要的也许是因为企业业务变更较为频繁所致，但技术的因素也不容忽视，尤其对于那些经常会“综合应用”各类财务报表系统（Excel表格、专用财务系统、第三方财务集成软件）的公司更是如此，原始财务数据的来源本就已经千头万绪了，技术上还要“没有困难创造困难也要上”地打出一个个“隔断”（不过这也往往与项目实施中业务人员过于强调“业务习惯”的影响有关）。这些问题导致了企业财务数据管理中一系列挑战，主要表现在：

- 数据收集困难，不同来源的数据难以集成在一起；
- 数据校验工作量很大；
- 财务数据在与非财务数据综合分析的过程中，限于系统集成能力的影响，难以相互印证；
- 难以及时处理，过去12个月86%的企业遭遇到报表生成超期的情况。

尽管报告提到的都是面向财务报表系统的问题，但对IT人而言，类似的故事在每个行业都不鲜见（比如物流企业、电信企业等），问题反复出现，说明这不是一个企业的问题，更不是一种技术问题，它体现出软件技术创新方向选择的差异性：

- 一方面，在互联网领域各种新技术、新软件产品层出不穷；
- 另一方面，在服务企业应用领域，相应技术落后于业务创新。

尽管我们常常听到“软件技术日新月异”这样的表述，但面对“平的”世界明显速度不足，以软件开发为先导的技术创新应该逐步向以数据管理为中心的信息管理创新转舵，也就是说面对新的业务调整，我们首先考虑的不应该是这个功能如何做，而是应该想到它会产生什么样的数据，这些数据与既有数据的关系，怎么让他们一起服务于企业的业务发展，然后才是软件的开发过程。但要做到这点似乎软件行业准备不足，毕竟随着开源软件的兴起，软件开发的中间产品大大丰富，相应信息管理理论虽然丰富、而且大型商业产品也不少，但真正将这些动态信息管理理论落地为可执行的功能整洁的开源软件似乎凤毛麟角。

可以说，企业信息管理创新≠用了什么新技术（云技术也好、三网融合也好）、新开发语言，关键在于创新怎样更好地服务业务发展的问题。这点不仅对于传统行业的企业信息化适用，对于那些试图开拓新空间并长期运营发展的互联网企业同样适用。

软件开发未动、数据管理先行。



王翔
软件架构师，主要研究方向为XML、.NET、领域设计和PKI应用。工作之余喜爱旅游、写作和烹饪。

系统架构与业务架构

系统架构师在SSH基础之上，应该再封装一个业务层的架构设计，这个业务层的架构设计是跟自己的产品和业务需求紧紧绑定的，是为业务服务的。

在2010年8月，我曾参与了一个业界知名人士主导的流通行业B2B2C产品的研发。从业务功能上讲，B2B2C中的B2B就是ERP、进销存功能模块的一部分，属于电商ERP。从2012年2月到现在，我又参与了一个知名家电企业ERP的研发，当然，我并没有参与ERP功能模块的研发，而是做了他们的ERP与淘宝、拍拍的接口设计。在这里，我将与大家分享我在做进销存、ERP这类产品时的体会，因为电商ERP也是ERP。

国内绝大部分企业，对于ERP、进销存这类业务系统的开发，采用的都是经典的SSH（Spring Struts Hibernate）架构。对于ERP、进销存这类业务系统而言，架构必须分层，分为系统架构和业务架构两层。看到这里，每个人都会说，我们一直是这么做的啊。但我观察到很多企业的业务系统，都没有业务架构层，直接拿Spring的系统架构作为自己的架构设计，由JSP（View层）调用Action，Action调用Manager层（所谓Manager层就是业务逻辑层），然后Manager层又调用跟数据库打交道的DAO（Data Access Object）层。

如果采用的是Hibernate，DAO又得调用Hibernate的类，然后每个表还得再生成一个BO（Business Object）类。这样，一个最简单的查询也得牵扯到Action→Manager→DAO→Hibernate→BO一共5层、5个类，外加一个JSP文件。Spring其实一点也不轻量，如果JSP再引入jQuery或者Ext这类视图层解决方案，一个简单的增删修改和查询，哪怕只有一张表，就得有10个类和一大堆配置文件。因此我认为这是ERP、进销存这类业务系统存在十几年却没有任何一个成熟方案的根本原因。采用原型开发、结对编程、极限编程、敏捷开发、持续集成都没有任何用，个人很偏激地认为，这些方法都是某公司拿过来骗人的。否则，ERP、业务系统也不会失败率那么高了（几乎100%失败，不上ERP等死，上ERP找死）。如果XP能像宣称的那样，拥抱变化，那么也就不会100%都失败了。

那么该怎么办？其实SSH框架并不是一无是处，我觉得它是做系统架构用的，使用对象是系统架构师，而不是程序员。系统架构师在SSH基础之上，应该再封装一个业务层的架构设计，这个业务层的架构设计是跟自己的产品和业务需求紧紧绑定的，是为业务服务的。

我还是举那个B2B2C经典的例子，一张采购入库单，触发了以下几个业务逻辑：（1）采购入库单本身主子表单据的增加；（2）采购商应付账款增加；（3）采购商库存在增加；（4）厂商或者分销商应收账款在增加；（5）厂商或者分销商会有一个销售出库单；（6）厂商或者分销商库存在减少。如果采用经典的SSH架构，这个Manager层得多复杂啊。为什么会有一个Manager层？高人们就会告诉你，一是因为这是一个事务，必须有总控，入口单一；二是因为要复用，可以拿出去给其他模块使用。对于第一个理由，我是同意的，但事务的起点，我也不认为是Manager层，而是Action触发的，我觉得Action才是业务逻辑的装配，而Manager层根本没有存在的必要；至于是否复用，我个人认为复用的是最小逻辑单元，而不是整个业务逻辑。比如，增减库存模块是可以复用的，而出入库单的业务逻辑，是没有任何可复用可能的。系统架构师的作用，就是把这段非常复杂的逻辑，设计出耦合性更小的业务层架构，给程序员使用，而不是把SSH简单包装一下，就直接扔给程序员了。折腾程序员的最终结果，就是跟装修一样，大家（开发商和客户）都不满意，产品或者项目开发周期延迟。



邢波涛
北京新软孚信息技术有限公司技术负责人。关注SaaS管理软件和B2B、B2C电子商务的融合。

风起亚洲公有云

Joyent技术在中国的崛起

随着移动互联网和云计算的兴起，社交网站（SNS）、电子商务网站、手机游戏等渐渐成为现代人生活中不可或缺的一部分。支撑这些应用的背后，公有云的作用不可小觑。在TechTarget中国发表的2012年全球云计算服务商排名榜中，除了有众所熟知的亚马逊等厂商外，还有一家未为中国用户留意的名为Joyent的公司赫然在列。自2004年起，Joyent就一直提供基础架构即服务（IaaS），只是那时这种服务还未被人称为“云计算”。实际上，社交网站LinkedIn早已使用其服务，每月推出数十亿页面，还有超过25%的Facebook应用程序也都运行在Joyent之上。能够吸引到如此多的Facebook开发者，该技术到底有何优异之处？中国的电子商务、社交网站或手机游戏开发者们是否也能使用到该技术呢？

答案是肯定的！联科集团（中国）有限公司于2010年与Joyent宣布成立战略联盟，推出名为“风起亚洲Fengqi.Asia”的公有云平台，向亚洲的云服务和游戏服务供应商提供云计算软件、服务和支持的整体解决方案。目前风起亚洲云在中国大陆、中国香港及新加坡均有成功案例。

风起亚洲云的SmartOS与同类热门云平台相比，性能优异。下表是基准测试结果。

风起亚洲云平台与其他公有云平台对比

类别	基准测试	风起亚洲超过：	
		国外某知名A云平台	国内某知名G云平台
磁盘I/O测试	IOZone：写	+589%	+392%
	IOZone：读	+1513%	+1228%
内存I/O测试	RAMSpeed：整型	+11%	+27%
	RAMSpeed：浮点型	+32%	+18%
CPU测试	SciMark稀疏矩阵	+73%	+174%
	SciMark稠密矩阵	+53%	+131%

事实上，风起亚洲云技术有累计超过8年的公有云服务经验，不间断地为数以千计的客户提供游戏、移动应用程序和网页托管等领域的服务。历经考验的云技术使得风起亚洲云能在众多IaaS服务提供商中脱颖而出，获得客户的青睐。简而言之，风起亚洲云有如下优点：

- Lean Virtualization技术——大大减少了虚拟化过程中的浪费，并把用户的资源放在应用服务上而非虚拟化上；
- 自动扩展的弹性计算——按实际使用情况免费和自动扩增4倍于平常的CPU资源（CPU Bursting），以轻松应对意外的流量高峰；
- 精准优化——独有的DTrace技术可对应用进行实时进程描绘，以使客户的应用能够优化云上的系统架构，帮助客户将每月数据存储和带宽的支出降至最低。

现阶段有些公有云厂商在提供IaaS服务时，仍存在诸多问题，例如将大量计算资源浪费在虚拟化上、数据流量等多种服务都要额外收费等。这样一来，客户产品性能提升和成本降低的幅度都很有限。而风起亚洲云提供面向实时Web应用的优化虚拟环境，与复制硬件的传统虚拟机相比，可大幅降低成本并提高应用性能。例如，风起亚洲云的一位新加坡客户，从事电子商务业务，需要5部计算机共240GB的内存来处理每秒5000次的并发访问。使用风起亚洲云后，只需6部虚拟机共24GB内存即可解决问题，总体成本降低了80%，应用扩展性也大大增加。此外，风起亚洲云还集成了Node.js开发及运行环境，可以大幅降低“实时数据密集型应用”的时间延误，而这是在社交网站、游戏、移动应用以及其他高性能计算领域中亟需关注的关键问题。

据IDC预计，中国IaaS市场将保持高速发展，到2016年其规模将超过10亿美元。包括中国在内，世界各地企业都在寻求技术创新，以更快地推出新产品和建立核心竞争力。基于Joyent技术的风起亚洲云提供了经过实践检验的云技术，可以立即满足网络游戏提供商、手机应用及电子商务网站等客户的需求，并提供高性能且可扩展的稳定计算能力。欲免费试用风起亚洲云，可访问联科集团专门为CSDN读者设立的链接：http://fengqi.asia/cn/csdn-dev.html。

创新与实践

第四届中国云计算大会观察

文 / 刘江

2012年5月23-25日，由中国电子学会主办、CSDN与《程序员》杂志协办的“第四届中国云计算大会”在北京国家会议中心隆重举行。三天时间内，来自全国各地政府部门、企业、科研机构、高校等单位的官员、CIO、CTO、技术经理、科研人员、软件工程师、信息系统运维人员、产品与采购经理等专业人士共8200人次参加了本次大会，一百多家国内外知名厂商在展览展示区亮相，趋势演讲和技术讲座逾百场。无论是会议规模还是参会人数，本届大会再次超过以往，毫无疑问将成为2012年度业界最受瞩目的IT盛会之一。



中国云计算落地：挑战与机遇

2012年被称为“中国云计算实践元年”，随着国家各部委对云计算的大力支持，各地方政府主导的云计算建设如火如荼地展开，阿里云、盛大云、新浪云、百度云等公有云平台的发展，腾讯、淘宝等开放平台的兴起，企业内部云解决方案的层出不穷，中国云计算切实地进入到落地与实践阶段。本次大会的主题也因此定为“发挥示范引领作用，推动云计算创新实践”。

对云计算在中国落地过程中的问题，与会领导有充分的共识。5月24日全体大会上，大会名誉主席、中国电子学会名誉理事长、原信息产业部部长吴基传在开场致辞中指出，在中国云计算如何落地这个问题上，对挑战与机遇要有清醒认识。中国云计算与世界先进水平还有差距，据国外的调查报告，全球云计算产业规模中，美国占了60%，而中国只占3%。同时，云计算的发展需要遵循科学规律，防止走简单、重复、浪费的老路。全国人大常委会副委员长严隽琪的主旨演讲强调了云计算标准化、海量数据、智能化技术的重要性。国家发改委高技术产业司副司长徐建平认为中国的云计算发展将是一个较长的过程，目前存在较大盲目性和信息安全隐患，需要在加强统筹规划的同时，发挥市场导向作用。中国电子学会理事长和陕西省常务副省长娄勤俭在讲话中也指出了国内云计算发展中重复建设、建设与需求脱节等问题。

但云计算对于中国更意味着巨大机遇，政府、企业、科研机构和广大用户应该通力配合，拥抱这次大变革。吴基传相信，云计算将大大改变商业模式和行业生态，它能够通过大规模的公共服务，带动上下游企业和服务企业的技术创新和产业发展能力。在技术、市场及政府的政策大力推动下，中国云计算必将迎来一个快速发展的新机遇期。严隽琪指出，普及信息技



术、为其他产业提供新的商机，是云计算的使命所在。业界应争取机会在云计算操作系统、云计算服务等关键领域实现突破，打破长期以来我国在高端服务器、基础软件、核心芯片等领域对国外的路径依赖。工业和信息化部副部长刘利华介绍了工业和信息化部作为行业主管部门将进一步加强云计算规划引导、政策扶持和标准制定，加大科研投入和示范推广力度，加强对网络与信息安全的保护，不断优化我国云计算产业发展环境。娄勤俭也认为，云计算代表着信息化和工业化相融合的大趋势，是一种现实的必然，需要政府通过制定中长期和近期的产业发展规划，鼓励引导社会各领域以云计算为基础发展信息化。

李德毅院士的演讲在定调“云计算已经成为当前信息通信行业的主旋律”之后，阐述了大数据在云计算中的重要性、社区众包对软件开发方式的影响和位置服务与云计算结合的巨大潜力。他指出，安全问题其实不是云计算的拦路虎，云计算安全实际是信任问题，大众参与所形成的由下而上的民主力量将成为建立服务信誉最基本、最重要的保证。他还建议，政府对云计算的支持应该重点放在社会民生服务上，而不是为购买云计算数据中心的设备买单；云计算的地域性示范试点固然重要，但互联网本身没有地域性，更重要的应该是加强行业示范。他建议工信部牵头解决最后一公里的宽带瓶颈问题，树立“先有路，后有车”的观念。李院士的演讲博得全场热烈响应，掌声不断，并在微博上引起热评。

梅宏院士主要从软件技术发展的角度提出了自己的思考。他认为云计算实际上就是一种计算集中，不应是未来唯一的计算模式，技术体系上还缺乏系统性，在模型、软件质量以及评估等方面都有问题。他还介绍了互联网开放模式下的网构软件开发方法。李未院士同样聚焦于云计算大背景下的软件开发变革。他认

为云计算具有超、变、散的特点，产生了超量的信息系统，使传统软件工程发生困难。他就此提出了群体软件工程，并阐述了开发原则和框架。

谁将成为中国的Amazon

3月美国之行（参见本刊5月期《美国云计算印象》一文），Amazon在美国整个云计算产业链中的关键作用让我留下了深刻印象：几乎所有方面不是在努力与Amazon竞争，就是基于Amazon发展壮大。缺乏Amazon这样公认优秀的公共云服务商，是目前中国云计算产业链的最大问题之一。

谁将成为中国的Amazon？相信也是本次大会各位参会者心中都在思考的问题。

三大国有电信运营商显然在品牌、实力、用户数、资质、基础设施服务经验等方面具有先天优势。5月24日全体大会中国电信、中国移动和中国联通悉数出席，均由集团副总为代表做主旨演讲，可见对云计算的重视程度。战略上三大运营商比较趋同，均在整合内部原有资源、加大硬件和研发投入的基础上，同时建设个人云和企业云，推进云计算商用，规划很大。虽然美国运营商和传统IDC服务商中也有RackSpace和Terremark、AT&T转型比较成功，在公共云市场上与Amazon抗衡，但中国的运营商必须破解体制困局、IT服务短板和技术储备上的先天不足，这并不容易。中国电信今年专门以新体制成立云计算公司，值得关注。

技术储备恰恰是同台演讲的华为和中兴等国内知名电信设备厂商的优势。在传统电信领域，爱立信首倡的服务托管模式其实已具备相当的私有云特征，华为和中兴当然不会放过这样千载难逢的进入更大企业IT市场的机遇。而且，通过为全球多家运营商提供公共的私有云服务，它们还有可能成为向公众提供服务的平台。当然劣势也是非常明显的，它们需要迈过至少三道坎：从电信到IT，从2B到2C，还有互联网化。

阿里巴巴、百度、腾讯、新浪、京东等为代表的互联网公司呢，它们会将Amazon的成功搬到中国吗？在技术和基因上，它们确实更有机会，考验它们的是远见、执行、资金和耐心。

而对于微软这样的国际平台型公司，牌照、监管和本土化运营是最大的软肋。毕竟至今还没有国外互联网企业在中国取得真正意义上的成功。

VMware、IBM、浪潮、CA、Intel、Dell等国内外IT厂商应该没有成为中国Amazon的想法，而它们面临的挑战却一点也不少。Facebook、淘宝等互联网厂商正在将研发能力延伸到服务器、存储和网络硬件领域，它们与低功耗芯片、OpenFlow、高效能服务器的创业企业正在共同打造全新的软件定义数据中心。

谁将成为中国的Amazon？让我们拭目以待。

谁将成为云时代的Linux

本届大会上，OpenStack、CloudStack和Hadoop等开源技术平台极受关注。由于事先估计不足，5月23日的开源云计算平台技术专场大大超员，致使许多持票参会者无法进场，可见其火爆程度。

开源云平台技术中，大数据处理的利器Hadoop是目前实际运用最为广泛的。而OpenStack和CloudStack、Eucalyptus都是IaaS层的云平台技术。前者的优势是开源早，社区活跃，而且得到了诸多重量级硬件公司的支持。CloudStack技术成熟，但开源不久，支持不足。Eucalyptus与AWS API兼容，而且得到了Amazon的支持。Cloud Foundry是PaaS层的平台，与其他各平台都可以共存，支持广泛的编程语言、框架，而且有VMware的大力投入，发展也非常迅速。它的竞争对手是Red Hat的OpenShift，但后者羽翼未丰。

虽然目前各平台不是完全直接的竞争关系，但随着平台扩展，发展最好的将很有成为统一平台。Cloud Foundry研发伊始就毫不掩饰自己成为21世纪Linux的决心。而Hadoop虽更多用于离线分析，但在活跃社区的添砖加瓦下，也开始向在线计算扩展。本次大会上Cloudera的CTO Awadallah的演讲题目就叫现代数据操作系统。IaaS层向PaaS扩展更是时间问题。

它们中谁将成为云时代的Linux，是云计算发展的焦点之一。

第四届中国云计算大会

精彩语录



云计算产业发展主要通过企业的自主研发，按照用户需求，提供用户服务。社会化服务指云计算最终要能够像供电供水系统一样，把计算资源作为一种公共基础设施，通过整合网络上的计算资源，来高效率、低成本地满足社会和个人对信息的需求。

——中国电子学会理事长 娄勤俭



由软件定义的数据中心要具有所有硬件的能力，且将这些硬件能力作为软件服务来提供。这也包括如何实现自动化管理，以便能够有机地、自动化地将所有的服务结合在一起，提供统一的接口以方便使用。

——VMware全球首席技术官兼研发高级副总裁Steve Herrod



安全不是云计算的特殊问题，很多信息系统最大的问题都是安全，特别是对于重要信息系统来说，安全一定是第一位。由于云计算实现了资源的高度整合，规模很大，如果云计算系统出了问题，它的影响远远超过传统信息系统，所以应毫无疑问地将安全放在第一位。

——中国工程院院士 倪光南



IT架构在很多年都会是一个混合的架构，因此数据、系统和应用的兼容和平滑过渡，将会是一个重要的任务。

——微软公司全球资深副总裁 张亚勤



我们对云计算的看法有四个层面：网络为基础，计算和存储为核心，安全为保障，应用为导向。

——华为IT产品线副总裁 朱近之



云计算本质是用非常先进的复杂计算和存储能力将资源整合在一起。最好的应用是需要大量的数据存储计算和分析应用。同时，云计算能够服务移动性需求，并且能够非常弹性地利用资源。

——百度网讯科技有限公司执行总监 范丽



建议五个云计算试点城市的宽带战略目标定为100Mbps，重点区域为1Gbps，全城Wi-Fi覆盖。要让试点城市市民如同享受公交一样享受宽带，试点城市确保每户家庭1TB免费存储。家庭是城市的细胞，有了智慧家庭，才有智慧城市！

——中国电子学会云计算专委会主任委员 李德毅



移动互联网时代，智能终端不断普及，网络带宽不断增加，为业务创新带来了极大的发展空间。而移动终端的尺寸和耗电量限制了其计算和存储能力，难以为客户提供更为复杂的应用；网络上海量的访问日志难以保存和处理，难以进行数据挖掘和优化。而云计算的出现将弥补这些不足。

——中国移动通信集团公司副总裁 李正茂



我们想把工作放到云上时，需要对其进行隔离、规约和加密。所谓隔离就是某种机制把所有的客机分离，当然，这就需要一个虚拟的监视器来区分这些客机。它只能通过一台虚拟机进行输入输出，才能保证完全的独立和隔离。

——英特尔云战略与规划总监 Billy Cox



信息量和信息投入在增加，但信息封闭和数字鸿沟并没有缩小，原因有两点：一是IT基础设施仍不够廉价，无法提供普遍的服务；二是即使把价格降低到能够承受的范围内，封闭和隔离造成信息孤岛仍然很难消除。

——中兴通讯首席架构师 罗圣美



在某种意义上，云计算就是一种新的集中计算。这种集中计算带来的好处是成本能够被分担、维护简单。但云计算不应是未来所有计算的唯一模式，个人计算、组织内部计算仍然是有必要的。

——中国科学院院士 梅宏



现有的很多云计算平台只适用于某一种特定的硬件和组织系统，我认为这并不能被真正地定义为云平台，通用是云平台最核心的要求，即可以把软件部署在不同的硬件和系统上，适应不同的应用。

——友友系统CEO 姚宏宇

大会更多内容请参见官网： www.ciecloud.org/2012

灵活、开放、自由的统计语言

第五届中国R语言会议剪影

文 / 高涛



R语言的应用现状

R语言是一门专注于统计分析及绘图的编程语言。与其他众多流行的编程语言不同，R语言最初并非被设计成一种通用的编程语言，而只是专注于数据分析和统计计算。因此，R语言最初只是在高校统计专业教学当中使用，然而，随着R语言在国内的推广和普及，以及工业界对数据分析和可视化的需求越来越大（如淘宝、京东等公司），R语言不仅在国内高校广泛所用，它也正在快速地向工业界进行扩展。

在R语言的传播和推广中，一股不可忽视的力量是2008年以来每年一届的中国R语言会议（简称China-R）。China-R的宗旨是促进R语言在中国的推广和发展，并为各领域的R语言使用者提供一个分享和交流的平台。目前China-R已经开设了北京和上海两个分会场，分别在每年的5月和11月召开。

精彩议题介绍

2012年的5月26日，第五届中国R语言会议在中国人民大学举行。会议共进行了18场专题报告，内容涉及数据挖掘、数据可视化、高性能计算、统计模型算法等热门话题；参会人员来自全国高校、研究所、公司企业等100多家单位，总人数将近300人。

本届会议主题如下：

R漫谈、基于R和pentaho的全套开源BI平台的实现、个性化推荐的十大挑战、30分钟学会ggplot2、R中大规模数据的整理和分析、用bignmf进行非负矩阵分解、COS沙龙介绍、knitr与动态报告生成系统、统计辟谣训练营、R与金融投资分析的框架、R在精算中的应用、R在社会网络分析中的一些应用、Data Mining with R-ORE、Rcpp快速入门、R在工业智能中的应用、隐马尔可夫链模型、机器学习的试验设计初探、用R实现随机森林的分类与回归。

由于篇幅所限，下面将简述其中最受欢迎的3个报告。

李舰：基于R和pentaho的全套开源BI平台实现

本次会议的一大亮点是由Mango Solutions中国区数据总监李舰带来的邀请报告“基于R和pentaho的全套开源BI平台实现”。pentaho是世界上最流行的开源商务智能软件，包括BI服务器、数据整合工具、OLAP引擎、报表编辑器、数据挖掘等组件。由于R与pentaho都是开源的产品，因此它们之间可以很方便地创建接口进行连接。李舰展示了他自己编写的一个R软件包rpentaho，用来在pentaho平台下创建R的应用。利用R强大的绘图功能，用户可以在pentaho中创建高级的统计

图形，并生成动态的、可交互的界面。此外，当前R具有5000多个扩展软件包（仅计算了CRAN和Bioconductor），因此通过调用R的扩展包，pentaho的功能也将得到极大的扩展。作为示例，李舰展示了R调用Google Visualization API的方法，并将其整合到了pentaho之中。

肖凯：30分钟学会ggplot2

ggplot2包是R中新兴的绘图软件包，目前已被超过50个包引用。肖凯以太极剑法和ggplot2的相似之处入手，介绍了ggplot2包的绘图理念，包括ggplot2的基础（数据、映射、标度、几何对象、统计变换、坐标系统和图层）及相互关系。然后剥茧抽丝地展示了如何绘制基本统计图形，例如散点图、直方图、条形图、箱线图等，以及如何自定义图形。最后提供了丰富的学习ggplot2的网络资源。听完肖凯的演讲，让人认识到传统的绘图语句（包括R中的基础图形设备）拘泥于描点、画线、配色等细节的描述，而ggplot2则是在数据层面上考虑问题，基于Leland Wilkinson的先进统计语法，把图形拆分为一些互相独立并且可以自由组合的成分。打个比方，传统的绘图语句是一个字一个字地往外蹦，而ggplot2则是一句话一句话地说。

ggplot2的作者Hadley Wickham是R社区中的高手，他的代码模块化程度高并且容易读懂。除了ggplot2，他的reshape2、plyr、lubridate在处理数据方面同样非常高效。此外新兴语言Julia也受到了这些软件包的启发。

潘岚锋：用bignmf进行非负矩阵分解

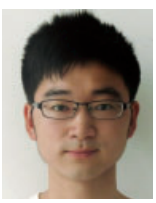
吴军在《数学之美》“矩阵运算和文本处理中的分类问题”一节中，讲到了奇异值分解（SVD）在文本挖掘中的应用。近些年来的研究发现非负矩阵分解（NMF）也有类似的功能，并且几何解释非常直观易懂、解释性更好，已在文本挖掘、图像处理中得到了广泛应用。R中有NMF包实现该方法，但计算效率非常低，而且其中使用的算法已有些过时。bignmf是潘岚锋近期正在开发的R包，其中使用他新提出的算法实现了非负矩阵分解，该方法在理论上比经典算法速度快很多。其核心代码是由C++实现的，外层则使用R程序调用，提升了程序的速度和易用性。

该包的最终目的是实现较大数据集的非负矩阵分解算法，目前算法可以处理稀疏矩阵，同时具有很好的可并行性。演讲还提到未来计划实现的功能，例如当数据量超过内存限制时，支持将大数据存放到硬盘上，一次从硬盘上读取少量数据进行运算，从而解决内存限制。最后，报告分别列举了一个图像数据和一个文本数据的应用案例：在人面部图像的数据上，算法能很好地把人的面部切割成多个有意义的部分；在微博等文本数据上，算法能从两类微博中提取各自的关键词，分析两类微博的共性和特性。

R语言展望

R是标准的数据分析和统计编程语言，由于具有灵活、开放、自由、共享等特性，它的成长非常迅速，已然成为了数据分析领域最热门的语言。

随着R社区的不断发展和壮大，R在数据分析中的应用已经如火如荼，在高校科研、互联网、金融等领域都全面开花，不过由于R计算效率问题，使得在业界大规模应用中遇到了瓶颈，因此，高性能计算也是最近开发的热点方向，这一问题正在逐渐得到改善。另外，R软件包数以千计，品质良莠不齐，如何对它们进行良好的维护也是保证R成为标准数据分析语言的重要因素。中国R语言会议秉承着开放、自由、交流、分享的宗旨，触及R语言最新进展，深入R语言应用，为国内各领域R语言使用者提供了一个良好的分享和交流平台。我相信，每一个参会者都会感受到R语言无处不在的强大和自由，而这正是推进R语言发展的活力源泉之一。



高涛

统计之都主站管理员和编辑，中南大学统计系专业学生。正编写R中距离测度学习软件包sdml。对机器学习、因果推断和数据可视化非常感兴趣。

开发者为王

苹果WWDC 2012见闻

文 / 梁永模



WWDC举行的会场：Moscone West Conference Center

WWDC为Worldwide Developer Conference的简写，是苹果公司一年一度的开发者大会。每年的WWDC的Keynote，苹果公司都会发布最新的软件和硬件产品。其深远的影响力已使得每年的WWDC都成为业界关注的焦点。

WWDC 2012于6月11－15日在旧金山Moscone 会议中心举行。本年度的大会日期并没有提前公布。在注册报名开始的前一刻，开发者们才得到邮件，提示入场券已经开始销售。一小时四十分钟后，所有五千多张入场券便销售一空。在未公布售票日期的情况下，入场券如此抢手，足可以看出WWDC的影响力以及开发者对大会的热情。

WWDC 2012的Keynote精彩纷呈，在Keynote上发布了iOS 6、Mountain Lion、带有视网膜屏幕的新款Macbook Pro等产品，中国元素也成为了苹果战略的重要一环。Keynote的内容是否满足了大众的期望众说纷纭，但不可否认，本次的WWDC Keynote又将给业界带来新一轮冲击，很多标准将得到提升，消费者将可以体验到新技术带来的新体验和更多的便利 。

由于大会Keynote内容已被媒体广泛报道，本文将介绍大会的一些细节，特别是我在这次大会中的亲身感受。

专业的会议组织，专业的开发者，专业的布道师

WWDC 2012历时5天，共有112个专题培训（Sessions），125个实验室（Labs），上千位苹果公司的工程师，为来自60多个国家的5000多位开发者提供各种信息和帮助。庞大的开发者团体，纷繁紧凑的专题培训和实验室，长达一周的时间，这些都对会议的组织者提出了很高的要求 。在5天的时间里，会议组织非常有序，开发者的素质也很高，自觉排队、互相谦让，听从组织者的安排。每一个专题培训都会有上千人听讲，但还没有发现在专题培训时手机声响起打断演讲的情况。专题培训的信息显示在会场的各个角落，苹果的工作人员着装统一，热情主动，为开发者提供各种便利，同时大会期间苹果发布了WWDC 2012的官方应用，所有的公告都是通过iPhone或iPad消息服务推送到每一个开发者的手中的，这样最大限度地保证了大家及时得到大会的重要信息。


苹果布道师展现了强大的实例


互动环节开发者挑战苹果专家

WWDC 2012大会一共有5000多人参加，除了第一天参加Keynote的媒体记者外，绝大多数都是专业的苹果开发者。大家聚在一起非常有共同语言，iPhone、iPad、Mac的组合是每个开发者的基本配置，Xcode是大家使用的开发工具，应用设计、性能改善等都是大家共同遇到的问题。这样专业、庞大的开发者团队对苹果的布道师们提出了非常高的要求。每一个专业培训的内容都是针对各个平台的最新技术，或者解决开发者遇到的难题的内容。开发者可以根据自己的需求参加相应的专题培训，也就是说在每一个会场听讲的开发者，大多都是讲座涉及内容的专家。可以想象听众对内容的要求是近乎于苛刻的。非常让人敬佩的是，这些苹果的布道师同样展现出了强大的实力，不但会议内容非常的有针对性，而且讲座的方式、语言的组织，以及表现力都给人留下了深刻的印象。因为讲座内容切实解决了开发者的问题，台下掌声和欢呼声不断。例如 Objective-C 语言的最新版本中，开发者不用再为每一个@property做一次@synthesize，并且推出了新的@literals，这些都极大方便了开发者的工作。此外，所有的专业培训都非常有一致性，很容易让人联想到Keynote的风格。每个专业培训有统一的时间（1个小时）、统一的界面、统一的流程。每个专业培训会以Overview开始，罗列之前存在的问题，进而介绍为解决这些问题而提供的新技术。之后会以每个子题目为单位进行讲解，包括理论展开、代码实例、Demo演示、总结等，非常简洁高效。每个开发者在专题培训中都能学到新的东西。每个专题结束时，都会附加说明相关专题布道师的联系方式、查询文档的方法，以及论坛的地址，并统一以“WWDC 2012”的标语结束。简单统一实效的专题培训，专业的工程师和开发者，专业的会议组织，成就了 WWDC这场世界上最好的开发者大会。

苹果对技术的专注，对自己产品的荣誉感

苹果对于技术的专注，对公司核心价值的苛刻遵循是出了名的。我在这次的大会中通过两个事例感受到了苹果的技术文化。

1. Eat your own dog food，中文翻译为吃自己的狗食，即自己要使用自己的产品，并对自己的产品有荣誉感。在大会开始之前，苹果发出邮件提示开发者下载iOS版本的WWDC 2012官方应用。这个应用里面有所有的专题讲座、实验室的时间表、所有会议室、便利设施的地图、开发者关注的专题讲座列表、推送的公告信息，以及官方上传的图片等内容。这种以自己产品为大会工具，提供给开发者的方式不但实用，且给人以非常亲切的感觉。不仅如此，很多专题讲座以此应用为例，进行讲解。例如在一个应用性能相关的专题讲座中，苹果的工程师演示了官方应用在不同的标签切换时，怎样解决反应速度过慢的问题。开发者可以切实看到前后两种代码实现方式在实际的官方应用中带来了截然不同的效果，一个普通



开发者在接受苹果工程师一对一的指导



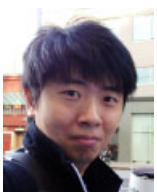
舒适的休息区

的WWDC 2012官方应用，能被如此利用，也真是让人叹为观止。

2. Stump the Experts。原来这只是一个晚上进行的附加娱乐节目，但节目内容让我感触颇深。这个节目基本上是开发者和苹果的专家互相提问和回答的竞赛，根据答题正确与否，最终决定哪一方胜利。据说这个节目已有21年的历史了。开发者提出了各种在我看来近乎变态的问题，但苹果专家的专注让我非常印象深刻。一个开发者拿出一个小零件，向苹果的专家提问，这个零件是哪家公司生产的产品，用在哪一代的iPhone上。几个苹果专家居然拿出了图纸，铺在地上，研究了一个小时，最终真的找出了这个小零件的出处，并说明这个零件是用在iPhone 3GS上的。另一个开发者提问，在OS X Lion中使用的最长的Selector名字是什么，同样，苹果的专家拿出电脑就开始找，经过一段时间准确地找出了答案。这种对自己工作的专注与荣誉感让我热血沸腾，这些专家对自己工作的自信，对保护公司荣誉的使命感，都深深地启发了我。

开发者为王的生态系统建设

在正式大会之前，还有各地区开发者聚会，席间苹果的一位开发者关系负责人说过这样一句话：“It's Worldwide Developer Conference, It's all about you”。苹果非常清楚，作为一个平台，开发者的辛勤劳动就是平台繁荣的关键所在。因此，开发者大会的方方面面都体现了开发者为王的原则。所有专题讲座都以解决开发者实际问题为出发点，竭尽所能了解开发者遇到的问题，着力减少开发者的开发成本。在所有专题讲座中，经常听到的两句话就是：“你只要用这几行代码就能实现这个功能；以后这些代码你再也不用去写。”除此之外，实验室的作用甚至要比专题讲座更为明显。开发者可以预约苹果工程师，得到一对一的指导。开发者有任何问题，都可以直接和苹果的工程师进行交流，得到最为专业的指导，解决自己的问题。此外，会场外提供的无限饮料、舒适的休息区、亲切实效的工作人员，无不体现了苹果对开发者的关怀。5天时间，开发者们得到了充分的时间与其他开发者或苹果工程师进行交流。这种专注决定了WWDC成为开发者获取信息，解决难题的最佳场所。



梁永模

百度高级研发工程师，从事百度移动云的技术布道以及SDK等工具的研发工作。曾就职于韩国三星电子，从事系统软件、iOS、Facebook等平台的开发。

异构系统架构编程的现状与展望

AMD副总裁Leendert van Doorn专访

文 / 刘江 卢鹄翔

未来如何更容易地编写异构程序？CPU与GPU是否将走向融合？6月22日，AMD副总裁Leendert van Doorn就异构编程领域的最新进展，接受了《程序员》杂志的专访。

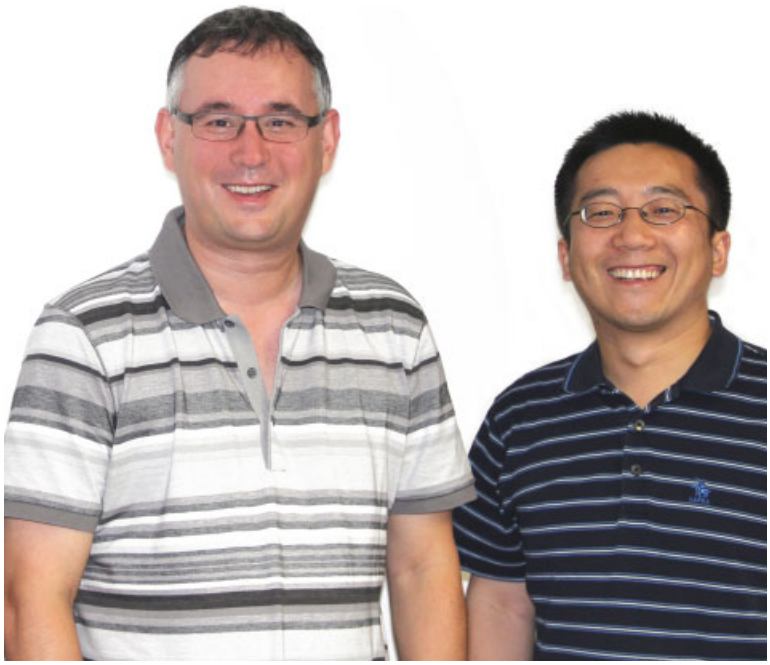
GPU加速将承担更多日常工作

《程序员》：请介绍一下你的工作。

Leendert：我是AMD院士，同时也是技术副总裁。在AMD公司，我负责具体将工程中的创新推向实际。我在这家芯片公司的软件部门工作，在这里与全世界软件行业的伙伴如微软、Google合作，向它们解释我们正在进行的工作，同时也了解它们的进展。

《程序员》：AMD同时制造CPU和GPU，在产品的发展路线上，与Intel相比有哪些不同？

Leendert：因为我们开发GPU，所以我们拥有强大的矢量运算技术，我们融合了CPU和GPU特性开发了APU。APU芯片不但具有很强的图形性能，我们更希望充分利用它的计算性能，这是个一石二鸟的好办法。在之前的演讲中我提到过一些应用，例如在生物学中研究基因和分子的化学构成、药物设计、地理研究等，这些领域都需要强大的运算能力。还有其他一些类似应用，尤其是自然用户界面方面的研究，令人激动。这些都是矢量运算处理器能够有所发挥的领域。



AMD副总裁Leendert van Doorn（左）与《程序员》杂志总编刘江（右）畅谈异构编程

《程序员》：目前已有一些最新的程序采用了AMD的GPU加速技术，例如Photoshop CS6、GIMP。除了这些图形方面的应用，在未来几年，操作系统是否可以直接利用GPU加速更多日常工作？

Leendert：我认为一定会。因为还有一些中间件，例如大数据处理也能利用GPU的潜力。我期望

看到的是，相对于操作系统，GPU不仅是作为一个外部设备，而是成为第一类实体（First-class Entity），最终由操作系统进行调度，决定将某类线程或者任务是应该分配到GPU还是CPU上执行，以达到最佳的运行效率。这也是我们希望实现的真正的异构。在操作系统内部，GPU擅长数据压缩，同时也擅长处理加密，这些都是操作系统中的基础功能。

苹果已在Mac OS X中加入了OpenCL支持，Google显然关注WebCL，而微软也在开发工具中提供了对异构编程的支持。这些工作还处在起步阶段，仍有很多工作要做。

HSA承载未来计划

《程序员》：OpenCL目前只支持使用类C语言进行编程，AMD是否计有划推广到其他编程语言？例如Erlang、Scala、Clojure原本已经对并行编程提供了很好的支持，如何才能使这些程序员也能从中受益？

Leendert：是的，AMD正在多个方面积极推动这些工作。一方面参与制定相应的标准，为其他语言开发者提供便利，例如利用Fortran进行OpenCL开发；另一方面我们也在寻求通过DSL解决，例如利用Aparapi，开发者可以使用Java语法编写程序，在运行时将Java字节码转换成可以在GPU上执行的代码。我认为上面你提到的编程语言都是我们未来会考虑的方向，但优先级还取决于商业因素，我们也在观察不同领域开发者的动向。

对于大多数开发者来说，并行编程难以掌握，我认为DSL是帮助他们参与GPU编程的完美方案，因为通过更高程度的语意抽象，可以隐藏这些细节。

HSA（Heterogeneous System Architecture，异构系统架构）是一种系统架构，它并不针对某种特定语言，也不仅仅支持OpenCL，尽管OpenCL在业界得到了广泛应用，这令我们很兴奋，但HSA毕竟是一项颠覆性的实践，我们还处在起步阶段，有许多工作有待完成。

《程序员》：除了程序设计语言，你是否认为其他与程序设计相关的软件，例如编译器，也会在未来进一步支持异构编程？

Leendert：我认为一个很好的例子是C++ AMP，微软在Visual Studio 2012中提供了对它的支持，对Windows开发者来说，这是一个强大的工具。另一方面，关于OpenCL，我们正与LLVM合作开发一些开源扩展和改进，我们也对将HSA作为一项开放标准充满信心。

《程序员》：前面提到微软在Visual Studio 2012中提供了对C++ AMP的支持，同OpenCL相比，两者有哪些不同？

Leendert：这么说吧，我们是否可以在C和Fortran之间做选择呢？答案是否定的。因为它们面对的是不同的客户、不同的应用领域。与此类似，如果用户需要在Mac OS X平台上进行GPU加速，C++ AMP无能为力；但如果用户在Windows上编程，C++ AMP是明智的选择，因为微软肯定会在这项技术上持续投入。

异构计算会是未来5年主要的方式

《程序员》：在演讲中，你展示了计算的三个阶段，第一个阶段是单核计算，第二个阶段是多核，第三个阶段是结合了CPU和GPU的HSA。那么在此之后，新的计算技术将会是怎样的？对异构计算来说，阿姆达尔定律又有怎样的影响？

Leendert：这是一个有趣的话题，我们也正在研究异构计算之外的架构，虽然目前GPU是AMD异构计算的重点，但异构并不仅仅代表GPU。例如利用固定功能加速器（Fixed-function Accelerator），或者FPGA进行OpenCL编程，这些形式也被称为动态计算或可重构计算（Reconfigurable Computing）。这些都是未来可能出现的新趋势，不过眼下还很难预料，但我认为，异构计算会是未来5年主要的方式。

软件公司总是希望程序编写以后能够运行尽量长的时间，少做修改。从这样的原则出发，对于历史遗留下来的代码，人们最关注的部分并不是怎样将程序并行。但未来我们的生活可能会发生诸多改变，例如我们无时无刻与互联网进行交流，医疗领域也可能会有翻天覆地的变化。

人们未来和计算机交互的方式，甚至人们对计算机的定义都会发生改变。这些新的应用为算法从根本上改变提供了契机，因为现有算法难以满足人们的需要。

这也就是阿姆达尔定律描述的情形——如果在CPU中执行程序的并行部分只有10%，计算架构上的进步也解决不了问题；对于GPU编程又是另外一种情况，人们一开始就知道它的架构擅长哪些工作，如果能从一开始就着手将99%的程序并行化，并对架构进行相应优化，情况就完全不同了。这也是未来激动人心的原因之一。

关于阿姆达尔定律描述的困境，需要依靠软件来解决。举例来说，如果在排序时使用冒泡算法，那么无论采用什么样的计算架构都很难将程序并行化，但如果使用归并排序算法，情况则大不相同。在针对GPU编写程序时，程序员从一开始就要考虑并行的问题。

GPU不会成为CPU

《程序员》：对于开发者来说，一种新的技术有一定优势是一回事，但很多时候只有当一项技术有杀手级应用出现时才会真正吸引他们采用，HSA的杀手级应用是什么？对于开发者你有哪些建议？

Leendert：我认为自然用户界面将会是HSA的杀手级应用，未来交互方式的巨大变化要求计算机提供更强大运算能力，我认为异构计算在这方面很有优势。对于所有的开发者，我总会建议他们至少尝试一下异构编程这种方式。不过每个人所在的领域和需要解决的问题不同，如前面提到的，对于很多人来说，使用领域特定语言或领域特定库是更好的选择。

《程序员》：Linux系统给人的印象是视频性能逊于Windows，例如游戏效果。对于HSA来说，这种情况是否会影响高端工作站或者计算机集群的性能？

Leendert：我认为人们感觉Linux或者Unix的图形性能不佳的原因应该归咎于X11，它是相当落后的窗口系统。与之相较，微软的DirectX则使应用程序可以直接与显卡交流。更先进的窗口系统是让人们摆脱这个印象的关键。对于HSA来说，内核模式驱动程序基本上只在配置阶段参与工作，对于HSA性能没有决定性影响。

《程序员》：有一种观点认为GPU正在慢慢向CPU的架构演化，比如缓存、乱序执行技术；另一方面CPU似乎也在借鉴GPU技术，比如AVX指令。未来两者是否有可能逐渐融合？

Leendert：这是一个很好同时也是很有挑战性的问题。尽管看上去GPU引入了这些CPU技术，但它最终不会成为CPU。在GPU中，这些技术仅以非常简洁的方式实现，因为一旦增加了复杂度，性能就会骤降。之前所有使用GPU的好处也会随之消失。

我们希望做到的是，从程序员的角度上看，GPU与CPU的兼容性越来越好，但在架构复杂度的角度上看，GPU永远不会变得像CPU那样。我们对于在CPU中引入AVX也很谨慎，一方面软件开发者真正采用这些指令尚需时日，另一方面x86处理器背负着沉重的历史包袱。即使加入AVX，能够获得的性能提升也未可知，而对于现有的产品，我们已经知道还有很多优化工作可做。因此我认为通过HSA比在CPU中加入GPU技术的可行性更好，HSA的意义在于如何让程序员更方便地使用所有计算资源，并且效率更高。

云计算基础架构：没有最好，只有最合适

IBM韩忠恒带你解读IDC《中国云计算基础架构建设指南》

记者 / 董世晓

毫无疑问，云计算作为这个时代的热点，已然走过了炒作概念的阶段，越来越多的中国企业开始着手进行云计算平台的建设。但与此同时，一个不争的事实困扰着它们：由于当前云计算仍处于初级阶段，所以企业并没有足够的经验自行进行云计算平台建设。

正因如此，6月15日，著名市场咨询和顾问机构IDC发布了它在云计算领域的重磅报告——《中国云计算基础架构建设指南》（以下简称“报告”）。报告通过实地访谈和案头研究两种方法，分析了中国企业IT基础架构现状与发展趋势，提出中国企业建设云计算基础架构的路线图，并就中国企业实施云计算基础架构项目的最佳实践给出建议，从而最终使用户在建设云计算基础架构时少走弯路。

作为在中国提供云计算基础架构建设的典型企业，IBM对这份报告有很多话要说。因此，在报告发布的同时，《程序员》杂志第一时间采访了IBM系统与科技部大中华区Power Systems总经理韩忠恒，请他就报告中提到的三个误区进行了剖析和解读。

误区一：云计算应该用分布式架构？云计算要用x86实现？

以Google和Amazon为代表的互联网巨头，其云计算采用的都是分布式架构和x86架构，并且取得了举世瞩目的成就，因此就往往给人一种错觉，认为云计算就是分布式架构和x86架构。但对于大部分企业级应用来说，做分布式计算就意味着要对应用进行代码级的修改，而异构平台情况在企业IT系统中普遍存在，且如果关键业务无法平滑迁移的话，也就无法实现真正的云计算。这导致的后果就是一部分核心应用仍然运行于传统平台上，另一部分应用运行于云计算平台，从而形成了云计算平台和原有传统架构并存的两个信息孤岛。此外，分布式和x86架构对于技术能力的要求并不低，对开发和运营管理的能力可能会更高，导致了成本依然居高不下。

因此，分布式架构只是云计算的一种模式，毕竟，分布式计算并非对各种应用都适用，目前更多地是用在互联网行业，而在非互联网领域尚未得到广泛验证。这时用户需要做的就是根据自身的行业特征来选择真正适合自己的云计算架构。

韩忠恒坦言，Power架构更多地用于企业级应用，比如金融、电信等私有云的建设中。在过去一年多的时间里，受到整个互联网环境的熏陶，一些这类传统企业纷纷上马分布式架构，但由于核心业务更适合集中式而非分布式，因此导致出现了两种套路，结果反而是浪费了资源。

误区二：无需虚拟化，只使用物理机一样可以实现云计算？

IDC在调查中发现，有些企业认为实施虚拟化只是实现云计算基础架构的手段之一，加之目前很多第三方虚拟化软件价格昂贵，甚至超出了购买物理服务器的价格。因此便打算直接使用物理机来构建云平台。

但在业务实际运行中，这些企业又发现，由于最小的资源单位只能是物理机，在很多情况下资源无法灵活地分配。同时，在资源分配时也往往需要IT管理人员干预，但殊不知，在增加人员工作量的同时也增加了人为出错的概率。可以说，整个架构完全没有达到自动化的目标。

韩忠恒介绍说，云计算是一个逐步深化的过程，企业级虚拟化是云计算架构建设的基础。当然，在建设过程中，除了虚拟化，企业需要同时考虑自动化、扩展性、可用性、安全性、可靠性等方面的问题，构建支持异构平台，而不是简单地创建虚拟服务器。通过对虚拟化管理实现自服务、自适应的基础架构，满足业务RAS（可靠性、可用性、稳定性）要求的企业级虚拟化平台是最重要的。而Power架构就为此树立了一个好


韩忠恒认为此IDC报告必将有益于中国云计算基础架构建设

榜样。首先，它提供了一个很好的虚拟化环境；其次，可以自动化管理，以达到资源池的良好体现和管理；再次，对架构进行融合；最后，达到自适应、自服务的云计算基础架构。

误区三：云计算会产生很多新型应用？云计算可以用来专门部署这些应用？

很多人认为在云计算基础架构上部署应用需要大量的底层开发工作，因此云计算会产生很多新型应用，或者干脆说云计算是为了部署新型应用而生的。对于这个问题，报告指出，很多企业应用的软件开发商并不是很了解云计算架构技术，实际上，云计算的实施不会创造出很多新型应用，企业依旧要依靠原有的业务应用来支撑业务发展，需要的大部分应用还只是传统应用的移植。

韩忠恒也对此进行了解释：“之所以出现这个误解，根源还是在对第一个误区的认识上。云计算作为一种全新的IT交付方式，用户希望能借助云计算改善IT管理，不只是处理新型应用，它们反而更希望传统的关键应用能够享受到云计算。作为云计算的一种形式，分布式架构与之前大相径庭，因此，肯定会带来新型应用。但就像前面提到的一样，分布式架构不是云计算的全部，新型应用也不会只存在于分布式架构上。新型应用可以从PC服务器开始，也可以从Unix开始，而且在中间还可能会进行转化。例如手机报这个应用，开始时选用的是PC服务器，但达到一定程度时，扩展就会出现瓶颈，就逐步向Power上进行迁移，从而实现更好的性能、更好的空间节省。”

总结

在总结此次IDC报告时，韩忠恒说：“事实上，人的感知和真正实施往往是两个反向过程，误区的存在正是因为缺乏实践，只有经过实践了，才能消除认识上的误区。企业在选择云计算基础架构时，需要综合考虑自身的业务特点和工作负载等信息，制定有效的企业云策略。就像IDC报告指出的那样，要经过‘规划、准备、实施、深化、应用和管理’这样的路线图，进行云计算基础架构的实施。我对此的总结就是，切记：没有最好的云计算基础架构，只有最合适的云计算基础架构。”

云计算基础架构建设一席谈

记者 / 董世晓

关于云计算基础架构建设，大家往往莫衷一是，各有论调。而已做了云计算基础架构建设工作的企业，有哪些经验值得我们学习？本文即邀请以下六位嘉宾，与我们畅享云计算基础架构建设的历程和经验。

金　剑	通一线信息技术有限公司信息技术首席顾问
陈春和	北大方正电子有限公司虚拟化与云计算解决方案总经理
杨洪波	携程旅行网高级架构师
冯宇彦	中国电子软件研究院云计算实验室主任
何少岳	东莞市东信网络技术有限公司CTO
申　健	星网锐捷网络技术有限公司信息资源部经理

促成开始进行云计算建设的因素

金剑：我们考虑的因素主要归纳为以下几点：第一，云计算是大势所趋，企业必须认清发展方向，以自己的优势占领先机；第二，目前云计算产业链虽初具规模，一些互联网巨头已谋篇布局，但中国市场尚需一段时间才能成熟；第三，目前几大巨头主要在公有云市场博弈，企业私有云市场还没有形成巨头垄断的现象，也就是传统行业的企业客户在市场上发现没有绝对优势的私有云产品后，就会把选择放得更加开放；第四，目前已经发展起来的云计算服务公司，如Google、Amazon、盛大云、阿里云等都是从自身的业务出发，将互联网或IT本身作为一种Business，它们提供的云计算服务也都是面向类似业务的公司，如电子商务、网络游戏服务商等；第五，传统企业更关心的是其业务运营，云计算只是作为一种IT技术能力为业务提供更好的服务，因此需要将云计算基础服务细节屏蔽并能提供更好的业务服务，这并不是公有云服务公司或单纯虚拟化软件等技术型公司擅长的地方。



金剑：传统行业的关键业务和数据库领域可能会采用Power架构

正因如此，我们看到了其中的机会，以对客户业务的理解、对云计算开放平台的研究和企业BI的经验，我们认为可以通过自身的实力、经验和产品为企业提供基于云平台的业务服务管理。

何少岳：我们公司拥有东信通、3G生活周刊、无线应用开发、移动广告平台等业务，各项业务彼此之

间有一定的交叉。近年来随着业务的不断扩大，原有的架构逐渐凸显出不足。综合考虑企业的业务需求、企业发展方向、人才结构、研发成本、运营成本等各方面因素后，我们开始着手云计算建设。

杨洪波：长期以来，我们都是自购服务器、自建运营中心、有自己的运维团队，现在也是。为了保证服务质量、维持服务的正常运行，我们服务器的峰值计算能力约是平均值的3~5倍，容灾备份机制中也存在诸多浪费，部分高端服务器达到顶配后还是不能满足旅游核心业务发展的需要。为了提高服务器的利用率，降低运营成本，我们开始考虑云计算建设。

冯宇彦：我们公司主要以开发行业应用软件、承接政府和企业信息化项目为主，有多个事业部开拓不同的行业市场。原来没有统一的数据中心，每个事业部都建立自己的小机房、购买自己的设备、招聘专门的运维人员。由于各自为政，导致在IP地址分配、机房改造和供电、制冷系统和运维上都存在着管理混乱、成本不可控、效率低下的问题。因此，我们需要建设集中部署、统一管理、专业运维的信息化系统。而云计算正是能够满足我们需要的最佳业务模式和技术架构。因此，我们决定把利用云计算的理念建设新的信息化系统。

申健：我们主要考虑以下五个因素：第一，我们很早就开始使用Salesforce的CRM系统，算是国内很早触云的企业，作为SaaS的用户我们感受到了云计算为企业带来的便利；第二，近两年云计算行业成熟度高，各大互联网企业有很多成熟应用案例，开源项目也越来越多，可以很方便地根据企业要求自行定制；第三，传统企业技术架构很难在低成本的前提下满足高可用性，借助云计算技术可以很方便地在不增加原有硬件的情况下大幅提升可用性；第四，计算和存储能力的弹性不够，无法保障突发压力，如果对所有的服务都做冗余的话资源使用效率又会太低；第五，出于TCO和节能考虑。

x86在云计算架构中遇到的问题

金剑：x86架构在人才培养、总体成本方面是具有


陈春和：Power并不像想象中那么高端不可接近

优势的。但x86架构在大规模部署后，由于单台服务器的性能较小，依靠大规模来实现计算能力和存储容量的扩展，就会带来以下问题：

第一，线性扩展，在较小规模状态下可简单增加服务器来扩展总体容量和计算能力，但规模扩大后，增加服务器并不能使计算能力按线性模式增长；

第二，动态扩展和动态分配，增加服务器后在不影响现有服务状态的情况下使平台能动态计算出新的总体容量和能力并不是一件容易做到的事情，同时负载均衡服务也需要将新的请求动态的传递到新增加的服务器上；

第三，管理复杂，规模增大后，可能需要跨越多个数据中心，这会带来数据异地存储、数据同步、状态更新、以及管理人员水平参差不齐等复杂问题。

何少岳：我认为x86架构在云计算的建设中主要存在智能维护、自动化部署、数据容灾及恢复、多核CPU资源利用、总体成本加大等问题，但归根到底还是人才的培养问题。例如，x86架构的系统如何能把系统性能发挥到极致？首先得让平台能合理利用各种系统资源，这就需要在构建平台时选择的语言能有效发挥多核CPU的优势，例如采用Erlang。

杨洪波：利用x86架构提高整体计算能力过程中，一个显著的问题是总体拥有成本，我们需要分布式存储技术和服务器的动态资源管理技术，这些技术



杨崇波：系统选型时首先要考虑企业业务发展所处的阶段

的引入从系统性层面看提高了整体拥有成本，对人才管理方面也提高了要求。我们前期先在非关键业务和纯技术性系统中使用，目前已经积累了一些经验，锻炼了队伍，逐步朝在线旅游核心业务扩展。

陈春和：在企业发展过程中，都有一个从分散到集中的过程，特别是大中型企业，它或多或少遇到过这个问题。例如，集中化后我们自有运维的就有很多系统，有些是自建的数据中心，有些是托管的，但随着业务的发展，数据中心里面的设备越来越多：一方面是自有数据中心的空间问题、能耗问题，所要面临的也是传统数据中心需要面对的困难；另一方面是企业采用云计算架构过程中对于专业人才的依赖问题。随着业务的发展，创新的力度加大，这时如果还是按照传统的方式去推进，便远远满足不了业务的发展。例如我们新建一个系统，按传统方式立项、采购、安装，实际上这样资源的利用率不高，而且这样一个繁琐、重复的过程，使得我们对业务创新的力度支持不够，有时可能商机也会因此而丧失。在这种情况下，企业往往会尝试着用虚拟化的方式来解决，包括服务器虚拟化、桌面虚拟化，这样在做的过程中间可以解决前面所说的问题。但在实施的过程中对我们的技术、包括掌控能力都是一个考验，所以如果企业也是用循序渐进的方式来进行的，例如最初我们避开关键业务，在一些开发和测试平台上用虚拟化的方式逐步推进。x86架构下的不稳定性以及云计算架构的复杂性，也带来了对技术人员的更高要求。在这个过程中，我们也培养自己的队伍来了解云计算技术，以便在今后真正的推进过程中遇到一些问题能自己应对。在此过程中也增加在人才培养这块总体成本。

企业在使用x86云计算架构时，难免会有迁移原有业务的过程。迁移过程中遇到的问题比较多，此时新建一个系统可能会比较好，因为如果我们要把旧系统迁移到云计算平台，一旦考虑到业务的连续性，就要制定完备的迁移计划。而且在最初的过程中可能使用不了，例如在迁移过程中有时还需购置一些新设备，然后才能把旧系统往上面迁移。在这个过程中我们对它的认知度有个过程，因此在最初迁移时要避开一些非关键性的业务，一步步来做。

在企业转向云计算架构阶段时，有的可能是大化小的方式，有的可能是小的组成一个大系统的方式，这需要根据自身情况选择，而最终选择哪种主要取决于自身的技术队伍。

此外，在安全管理方面，云平台的安全控制跟以前是不同的，需要制定一些措施，包括对数据的隔离和控制都得要一并考虑。

冯宇彦：x86架构为了增强处理能力，在数量上比传统小型机要多很多，相应带来了一些管理和稳定性方面的问题。例如，管理方面必须整合第三方的计算、网络和存储管理软件，系统的复杂度增加；另外，由于x86架构服务器系统的稳定性不如小型机系统，因此需要有完善的备份和容错方案。以上这些问题都有相应的解决方案，但需要第三方来整合，这也是目前用户没有蜂拥而上进行云计算转型的一个原因。

申健：在我看来，现在的架构已相对成熟，而且可供选择的架构方案也非常多，企业可以根据需要和实际情况，扬长避短，来决定自己的架构选型。个人认为早期x86架构的主要问题在于空间和能源的利用率不高，现在已出现很多高密度x86架构服务器，很可能未来中小型传统企业只需要一个标准机柜就可以构建出支持企业所有业务需要的私有云服务。

云计算带来的改变和收获

金剑：不同的客户采用云计算的目的不同，一定要了解云计算客户的原始诉求。

对以互联网为业务的初创公司，通过互联网购买公有云计算，可降低初始IT设备成本和运营风险。

对传统行业公司，尤其是大公司，初始考量是为支持业务应用，提高应用的可用性和可靠性，虽然虚拟化部分达到了这些目的，但云计算在此基础上更可提高管理的自动化和流程化。

何少岳：首先，引入了一批新鲜血液，参与到云计算架构的建设中，让团队更为健全和完善。其次，云框架的建设也让各项业务之间的资源按需高效使用，通过统一基础服务架构、授权开放接口的模式减少了部门之间的重复工作，提升了系统使用率和部署效率，降低了运维成本。

杨洪波：主要体现在两个方面：第一是服务器的自动化维护水平有很大提高，目前正朝自动化管理方面努力；第二是敢于攻克以前认为不好做、不敢做的工作——我们开始利用云计算技术做集约式的用户行为分析、开发业务API、日志集中处理、酒店业务系统监控。总体而言，总体成本没有增加的情况下，服务器的利用率得到了一定程度的提升。

冯宇彦：通过实施云计算建设，公司的信息化产生了如下改变：第一，信息化预算可以统筹规划，更加科学；第二，信息化系统统一采购、统一管理，大大降低了总体拥有成本和管理的复杂性；第三，新的业务需求可以快速满足，原来新项目需要几天调整设备、准备开发测试环境，现在只需要30分钟就可准备好开发测试环境；第四，事业部降低了运营成本，并且更专注于所在行业市场的开发，而不用分出精力进行信息化系统维护。

申健：目前我们最大的实质性变化是构建内部的PaaS，类似于SAE或Heroku。因为对原有系统做迁移和改造工程较大，所以效果也是逐步显现的。

目前已经在几方面取得了效果：第一，通过分层、冗余等架构调整实现了整个架构的高可用；第二，应用层和服务层可以水平扩展，提高资源利用率和架构的弹性；第三，通过HDFS、NoSQL等技术提供文件存储、索引、缓存等基础服务，减化开发工作并且可以进一步提升系统的可用性和可扩展性；第四，改变原来单机单系统的状况，实现了资源利用最大化，进而可以更快速部署更多的应用。我们在内部搭建了实验平台，用来引入各类最新的应用系统供企业内部员工试用。对于应用良好的系统通过动态扩展资源转为正式应用系统。

x86架构还是Power架构？系统选型应考量哪些维度？

何少岳：我认为最终的考量维度还是归根为经济成本和技术人才两个方面，公司可以选择最符合自身发展阶段、发展方向和业务需求场景的架构，毕竟适合才是最好的！

陈春和：IBM Power在部署速度、维护、能耗、可靠性上相比x86都有明显优势。x86虚拟机方案包含了虚拟化软件费用，所以要更高一些，而Power虚拟机方案中包含了IBM自家的PowerVM虚拟化软件，用户不需要额外购买虚拟化软件。在成本相差不多的前提下，Power虚拟机方案在空间占用、能耗、电力总成本上优势明显，更重要的是，Power平台还具有先天的稳定性优势，这对“将所有鸡蛋放在一个篮子”式的虚拟化应用来说很重要。Power确



冯宇彦：Scale out架构的技术难点和潜在问题是协同问题

实是值得推荐的一个平台，它并不像想象中那么高端不可接近。事实上，针对当前的云计算大环境，IBM Power积极投入其中，推出了丰富的Power平台解决方案帮助用户快速构架基于Power的云平台；为拓展行业，IBM发布“天工计划”，紧密联合ISV，为客户推出从应用出发的Power解决方案。综合这些因素之后，在选择系统的过程中，企业更应该相较于自身的应用服务需求和综合技术实力来考量。对于单一服务对计算性能和技术支持相对较高的业务，Power架构是不二的选择。但对于中小型应用规模的需求，以及缺少技术支持的中小型企业而言，x86架构也许是更合适的选择。

杨洪波：系统选型要考虑两个方面的问题。

第一是企业业务发展所处的阶段。一般分为创业阶段、早期阶段、中期阶段、中后期阶段和成熟阶段。创业阶段选型较容易；早期阶段的架构选型和系统选型需要为中后期阶段的业务发展做好铺垫，此时主体业务可能是x86的单部署结构，这样可以降低开发成本和人员管理成本，但部分系统已可以考虑采用x86的云架构，同时积累部分云计算的人才和经验；中期阶段，关键业务有一定程度发展，可以考虑Power架构部署，保证系统稳定，避免团队人员的快速扩张；中后期阶段，业务发展有了质的飞越，此时可以利用前面积累的经验，对系统进行云计算改造（一般而言，对电商类业务，技术团队规模可能到700~1200）；成熟阶段就需要继续考虑效率、成本、节能、人才管理等方面的平衡。



何少岳：x86架构的问题归根到底是人才培养问题

第二是企业业务的特性，涉及企业文化，包括企业的技术定位等方面。

金剑：一般在传统行业的关键业务和数据库领域还是可能会采用Power架构，但在与用户交互的Web和应用软件域可以采用基于x86的虚拟化和云计算平台，从而既可以保持架构的灵活性和可扩展性，又可保持数据和关键业务的可靠性。

冯宇彦：原来之所以Power架构和RISC架构无法替代，是因为它们在I/O吞吐量和稳定性方面有无可替代的优势，尤其是基于关系型数据库的关键应用，更是对这两个方面有很高的要求。但当前随着云计算底层技术的成熟，通过虚拟化管理软件，可以用实时迁移和多份拷贝来解决稳定性问题，而通过新型的集群存储和列数据库集群技术可以解决文件和关系型数据I/O大吞吐量问题。但毕竟这个迁移和转变不是一蹴而就的事情，开始时，企业可以先把Web和应用层系统迁移到Scale out架构上实施虚拟化及管理，将文件和关系型数据库放在传统的Power和RISC系统上，而后逐步尝试用新的数据库系统。

申健：系统选型可以从五个维度考量：TCO、成熟度（看成功案例）、可扩展性和扩展成本、运维成本，以及避免“技术绑架”，关注系统的开放性。

Scale up（纵向扩展）架构和Scale out（横向扩展）架构孰优孰劣？

金剑：对于互联网行业，Scale up架构的确遇到了一些问题，越来越多的基于互联网平台产生业务收入的公司开始采用Scale out架构，这是由于互联网的业务发展特点决定的。

传统行业的计算密集和关键业务大多采用Scale up架构，因为这类业务对IT在稳定性和可靠性方面的要求比对性能上的可伸缩性要求更高。Scale up架构在稳定性上设计更为周全，硬件质量和指令集更为可靠。但在扩展升级时会带来停机时间，虽然一

般会采用集群架构方式，但在架构管理上偏复杂，这是它需要提升的地方。

Scale out架构技术难点是网络上的数据同步和状态同步管理，尤其是跨数据中心的大规模IT基础设施对中国的南北网络未能完全互通情况下的网络延迟的处理，以及在大规模后的容量管理方面。

何少岳：Scale up架构在管理上相对来说容易些，不需要同时管理很多机器，但存在单点故障，会中断服务，存在大数据迁移和恢复的问题，资源也很有可能得不到充分利用。Scale out架构需要一个强大的管理系统来管理这些机器，这些机器有可能会跨几个IDC，维护管理是一个问题，无形中还增加了经济成本，但扩展性好，资源不足添加机器就行了，也不必要中断服务来做数据迁移等工作。但无论是Scale up架构，还是Scale out架构，数据隔离和数据安全都是一个具有挑战性的难题。随着近年来大数据的发展，如何充分考虑Scale up架构和Scale out架构的优劣去设计系统架构，是我们值得思考的方面。

陈春和：随着芯片业、制造业新技术的不断发展，Scale up架构相信还远远不到其瓶颈期。而且单就技术的优劣来看，Scale up架构似乎更能体现其技术的先进性。对于金融、证券等行业更多需要集中的计算资源需求而言，Scale up架构才是其目前最好的选择。

Scale out架构具有Scale up架构无法比拟的优势是其灵活的分布方式以及相对较低的进入门槛。但Scale out架构也有其潜在的问题——目前云计算架构的资源集中性，导致大量数据统一存放在单一地点，使得数据存储的稳定性问题以及难以容灾问题日益显现。而在云计算时代，存储和容灾对于架构的安全性问题起着至关重要的作用。

冯宇彦：Scale up架构发展到一定程度，受限于单体空间和架构限制，肯定在性能提升上不如以前快，不能完全称之为“瓶颈”。但Scale up架构肯定不如Scale out架构具备更大的灵活性和扩展性，这是它们真正的分别。这个发展趋势不是技术的原因，而是业务的原因，当前社会经济技术发展越来越迅速，商业竞争也日趋激烈，对信息化系统提出



申健：Scale up和Scale out有各自擅长解决的问题

灵活便捷和大规模横向扩展的要求是Scale out架构今天为人所关注的主要原因。不过，Scale out架构的技术难点和潜在问题是一个，就是协同问题，因为节点增加到一定规模，节点间协同通信的时延和数据量会降低系统效率，因此，采用Scale out架构的企业也应该根据自己的实际情况适当采用一些混合技术，如分区和分级。

申健：我是Scale out的支持者，对于架构本身其实不分好坏，只是用户能力、应用场景和系统平台不同。我认为对于中小企业来说Scale up架构的主要问题就是成本高、弹性差，中小企业业务变化快需要“小步快走”的模式，而Scale out正好满足了这种模式的要求。我想这也是为什么互联网公司大量应用Scale out架构，在互联网公司创业早期不可能投入太多的硬件资源，因为无法验证自己的商业模式和用户量，随着用户量的增加可以逐步增加硬件投入。Scale out的问题在于如果没有很好的前期规划，可能会随着节点的增加使维护工作呈几何数量的增长，空间资源的利用也会是一个问题，不过目前都已有相应的解决方案。从IT规划的角度看，Scale up是自上而下的规划方式，Scale out是自下而上的规划方式，有各自擅长解决的问题。



梁捷
UC（优视科技）技术总裁。1998年毕业于华南理工大学计算机专业，后长期耕耘于中国的电信和互联网市场，在电信及网络计算领域拥有超过10年的技术研发和管理经验。

梁捷专栏

Web App的未来

HTML5与Web App

最近一两年，HTML5和Web App的概念相当火热，吸引了很多人或多或少的关注。

从狭义上讲，HTML5是超文本标记语言（HTML）协议的一个升级版本。这个升级版本包含了一些新的元素，包括Web Workers、Web Storage、Web Sockets API、Server-Sent Events等。现在这个协议是由一个标准化组织——万维网联盟（W3C）来制定与普及的。

从广义上讲，HTML5是Web App相关技术的总称，包括HTML5、DOM API、CSS3、面向Web App的页面设计模式（如SPA）等（注：连接为：http://platform.html5.org/）。所有这些工作都是围绕着对浏览器的扩展。因此，从广义上来讲，大家把HTML5的含义进一步地泛化，现在涵盖到了Web App的范畴上来。

但说实话，当前HTML5太热了。人们在谈起它时，更多是在阐述一种概念，这有点像当年的Web 2.0。

Web App的核心优势

从现在来看，应用一般有三种：原生应用、混合应用和Web应用，如图1所示。

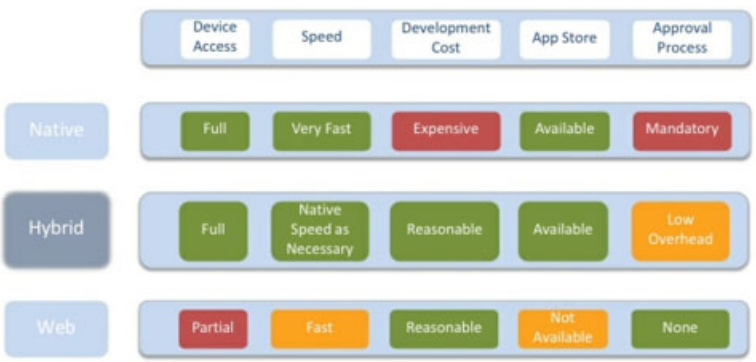


图1　应用的三种类型

相对于其他两种应用，Web App的核心优势有两点：一是开发模式的再次进化（对复杂性的封装），二是具有卓越的互联互通特性。

从开发模式来说，Web App的出现像是历史的再次重演。之前是从汇编到C语言，因为C语言相对高级一些，所以开发者真正需要做的事情就简单了很多。现在我们进一步从Java到HTML，也是把一些复杂的东西封装起来了。我们做应用不需要写边框、画界线，只需要用一个模板就能够起到界面显示的作用。

Web App的另一个很大的优势是有非常卓越的互联互通特性。最近两三年，网页游戏在游戏界非常流行，它就是依赖了Web形态应用的互联互通特性。很多流量能够很方便地导入这样的一个网页中，同时它也能够在页面内给其他应用导流量。这样卓越的互联互通性对产品的推广和营销都有很大的帮助。因此，在过去两三年里网页游戏得到了非常快速的成长。

现在很多开发者都感觉到，原来必须要用客户端技术去做的效果或者功能现在基本都可以通过Web App的相关技术实现了，通过这些简单的语言和配合，就能做到很酷炫的效果。例如UC应用中心里收录的豆瓣电台、3D游等一系列优质Web App。

标准的普及

一般来讲，W3C制定HTML5标准的过程是这样的：在经过大约五个步骤之后才能将标准正式对外发布（Recommendation），如图2所示。而2011年，HTML5标准刚刚走到了第二个流程，依旧处于工作草案状态。所以有人预言HTML5的普及还需要10年。但事实上，所有

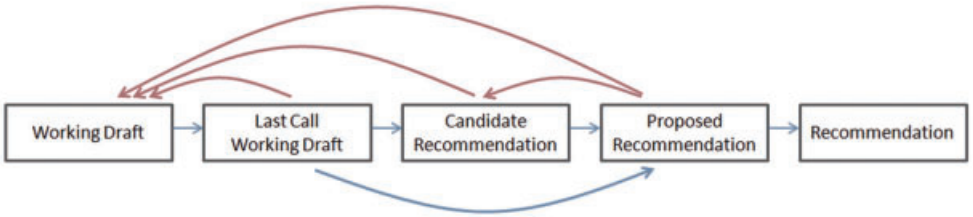


图2　W3C制定HTML5标准的过程

浏览器厂商都看准了这个趋势，在大笔投入。大家都已经在路上，唯恐落后。

作为首家加入W3C的中国互联网企业，UC非常认同HTML5的发展前景，也会积极推动这个标准的普及与发展。

首先我们关注HTML5的兼容性，因为我们发现很多开发者都把一多半的精力放在兼容性方面。兼容性首先要对标准全面支持。刚刚发布的Android平台UC8.5在兼容性方面做了很大提升，兼容性测试跑分为410+7，达到世界第一的水平。当然兼容性的解决也需要过程，最早推出U3内核时我们的跑分只有100多分。后来我们研发团队在这方面保持了长期专注投入之后才取得了这样的提升结果。

与兼容性同等重要的就是扩展性了，包括手机上各种设备能力的外拓。其实现在业界还没有统一的标准，所以UC就在推广这些标准的落地和实行。我们可以看到刚才分裂的状况，苹果不会去关心Android，而Google也不会去关注中国的一些山寨厂商。所以我们会站在中国手机用户的角度，去把刚才看到的那些和好处切切实实地落地。我们希望能够多做一些务实的事情。

同时我们也基于自身的业务需求，利用HTML5来提高展现能力。例如在UC资讯中心改版中，我们采用单页面应用设计思路，利用我们自己设计的ELF模块化加载器，前端搭载Zepto，使用基于事件驱动的MVC框架完成了项目展示。新版的UC资讯中心也使用了HTML5标准中的Web Storage和Application Cache。

此外我们还在2011年9月发布了国内最早的移动Web App应用中心——UC应用中心，希望能给优质HTML5应用提供一个很好的展现平台，也让更多的用户可以接触到这些优秀的Web App。目前UC应用中心已经收录了超过



图3　UC应用中心

300款应用，月活跃用户过千万，应用添加次数超过5000万次。

Web App的发展趋势

HTML5的蓬勃发展跟移动互联网的发展是密不可分的。过去几年，无线上网的设备数量都在翻番，但终端、操作系统等平台的差异非常大，开发模式、开发语言等都也有不小差异。HTML5的出现会是移动互联网的一把万能钥匙。因此，Web App很可能首先在移动平台取得较好的发展。

从技术角度看，以下三个特点会是Web App未来发展的三个主要趋势。

- 网页应用化
- 支持离线使用
- 支持调用手机硬件接口

网页应用化和支持离线使用会让Web App的能力得到进一步的拓展。还有就是支持调用手机硬件接口，例如照相接口，除了对焦、光圈、快门之外，还包括白平衡、调色等。如果WebApp可以支持的东西全都弄出来之后，Web开发者会觉得很幸福，因为可以用非常简单的方式做出非常专业的应用。

未来，基于HTML5的Web App将成为手机上的主流应用形态。我们希望UC可以帮助Web App的相关标准的应用和普及。我们也会致力于将UC打造成对Web App支持最完善的一体化服务平台。



智能算法

智能应用每时每刻都在影响着我们的工作和生活，然而对于许多软件开发人员来说，构建智能应用的技术——智能算法依然披着神秘的面纱。一方面，这些技术的潜在价值可以带来巨大的经济回报，商业公司往往不会轻易示人；另一方面，几乎所有的相关技术都源自学术研究，倘若缺乏引领，则难以深入其中。本期封面报道，我们将从搜索技术、社交网络数据挖掘、机器翻译、推荐引擎等角度，结合具体应用，为你揭开智能算法的面纱。

地图和本地搜索的最基本技术

有限状态机和动态规划

文 / 吴军

2007年，第一次在谷歌黑板报上写“数学之美”系列时，本地搜索和服务还不是很普及，智能手机不仅数量有限，而且完全没有结合本地信息。地图服务的流量与网页搜索相比，只是众多垂直搜索的一部分。今天，本地生活服务变得越来越重要，而确认地点、查看地图、查找路线等等依然是本地生活服务的基础。本文将围绕地图和本地生活搜索，介绍有限状态机和动态规划。

2008年9月23日，Google、T-Mobile和HTC宣布了第一款基于开源操作系统Android的3G手机G1。这款手机的外观和体验远不如一年之前苹果推出的第一款iPhone，价钱也差不了太多，但依然有不少人使用。它的杀手级功能是利用全球卫星定位系统实现全球导航。卫星导航的功能早在2000年前后就已有车载设备使用，但售价昂贵。2004年我购买的一款麦哲伦便携式导航系统，价格在1000美元左右（现在只要两三百美元），后来一些智能手机也开发了这个功能，但基本上不可用。Android手机的这个功能当时已经完全可以媲美任何一个卫星导航仪，加上它的地址识别技术（采用有限状态机）比任何一个卫星导航仪严格的地址匹配技术（不能输错一个字母）要好得多，结果麦哲伦等导航仪制造公司在G1发布当天的股价暴跌四成。

智能手机的定位和导航功能，里面的关键技术只有三个：第一是利用卫星定位，这一点传统的导航仪都能做到，不做介绍；第二是地址的识别，在本文第一节中介绍；第三，根据用户输入的起点和终点，在地图上规划最短路线或者最快路线，在本文第二节中介绍。

地址分析和有限状态机

地址的识别和分析是本地搜索必不可少的技术。判断一个地址的正确性同时非常准确地提炼出相应的地理信息（省、市、街道、门牌号等）看似简单，实际上很麻烦。例如腾讯公司在深圳的地址，我收到的邮件和包裹上面有如下各种各样的地址：

广东省深圳市腾讯大厦

广东省518057 深圳市南山区科技园腾讯大厦

深圳市518057 科技园腾讯大厦

深圳市南山区科技园腾讯公司

深圳市南山区科技园腾讯总部 518000（估计不知道准确的邮编）

广东省深圳市科技园中一路腾讯公司

……

这些地址写得都有点不清楚，但邮件和包裹我都收到了，说明邮递员可以识别。但如果让一个程序员写一个分析器分析这些地址的描述，恐怕就不是一件容易的事了。其根本原因在于，地址的描述虽然看上去简单，但它依然是比较复杂的上下文有关的文法，而不是上下文无关。

例如下面的两个地址：

上海市北京东路xx号

南京市北京东路xx号

当识别器扫描到“北京东路”时，它和后面的门牌号是否构成一个正确的地址需要看它的上下文，即城市名。上下文有关文法的分析既复杂又耗时，它的分析器也不好写。如果没有好的模型，这个分析器写出来很难看不说，还有很多情况覆盖不了。例如这样的地址描述：

（深圳市）深南大道和南山大道交口西100米

所幸的是，地址的文法是上下文有关文法中相对简单的一种，因此有许多识别和分析的方法，但最有效的是有限状态机。

有限状态机是一个特殊的有向图，它包括一些状态（节点）和连接这些状态的有向弧。图1是一个识别中国地址的有限状态机的简单例子。

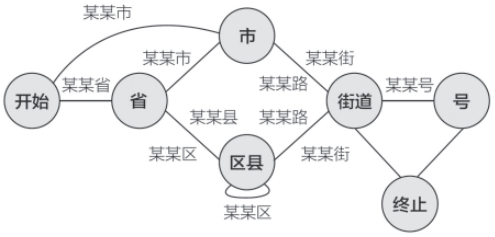


图1 识别地址的有限状态机

每一个有限状态机都有一个开始状态和一个终止状态，以及若干中间状态。每一条弧上带有从一个状态进入下一个状态的条件。比如，在上图中，当前的状态是“省”，如果遇到一个词组和（区）县名有关，就进入状态“区县”；如果遇到的下一个词组和城市有关，那么就进入“市”的状态，如此等等。如果一条地址能从状态机的开始状态经过状态机的若干中间状态，走到终止状态，那么这条地址就有效，否则无效。例如，“北京市双清路83号”对于上面的有限状态来讲有效，而“上海市辽宁省马家庄”则无效（因为无法从“市”走回到“省”）。

使用有限状态机识别地址，关键要解决两个问题，即通过一些有效的地址建立状态机，以及给定一个有限状态机后，地址字串的匹配算法。好在这两个问题都有现成的算法。有了关于地址的有限状态机后，就可以用它分析网页，找出网页中的地址部分，建立本地搜索的数据库。同样，也可以对用户输入的查询进行分析，挑出其中描述地址的部分，当然，剩下的关键词就是用户要找的内容。例如，对于用户输入的“北京市双清路附近的酒家”，Google本地会自动识别出地址“北京市双清路”和要找的对象“酒家”。

上述基于有限状态机的地址识别方法在实用中会有一些问题：当用户输入的地址不太标准或者有错别字时，有限状态机会束手无策，因为它只能进行严格匹配。（其实，有限状态机在计算机科学中早期的成功应用是在程序语言编译器的设计中。一个能运行的程序在语法上必须是没有错的，因此不需要模糊匹配。而自然语言则很随意，无法用简单的语法描述。）

为了解决这个问题，我们希望看到可以进行模糊匹配，并给出一个字串为正确地址的可能性。为了实现这一目的，科学家们提出了基于概率的有限状态机。这种基于概率的有限状态机和离散的马尔可夫链基本上等效。

在20世纪80年代以前，尽管有不少人使用基于概率的有限状态机，但都是为自己的应用设计专用的有限状态机的程序。20世纪90年代以后，随着有限状态机在自然语言处理上的广泛应用，不少科学家致力于编写通用的有限状态机程序库。其中，最成功的是前AT&T实验室的三位科学家，莫瑞（Mehryar Mohri）、皮耶尔（Fernando Pereira）和瑞利（Michael Riley）。他们三人花了很多年时间，编写成一个通用的基于概率的有限状态机C语言工具库。由于AT&T有对学术界免费提供各种编程工具的好传统，他们三人也把自己多年的心血拿出来和同行们共享。可惜好景不长，AT&T实验室风光不再，这三个人都离开了AT&T，莫瑞成了纽约大学的教授；皮耶尔先当了宾夕法尼亚大学的计算机系主任，而后成为Google的研究总监；而瑞利直接成为Google的研究员。有一段时间，AT&T实验室的新东家不再免费提供有限状态机C语言工具库。虽然此前莫瑞等人公布了他们的详细算法2，但省略了实现的细节。因此，在学术界，不少科学家能够重写同样功能的工具库，但很难达到AT&T工具库的效率（即运算速度），这一度是一件令人遗憾的事。但近年来，随着开源软件在世界上的影响力越来越大，AT&T又重新开放了这个工具的源代码。有限状态机的程序不是很好写，它要求编程者既懂得里面的原理和技术细节，又要有很强的编程能力，因此建议大家直接采用开源的代码就好。

全球导航和动态规划

全球导航的关键算法是计算机科学图论中的动态规划（Dynamic Programming）的算法。

在图论中，一个抽象的图包括一些节点和连接它们的弧。如果再考虑每条弧的长度， 或者说权重， 那么这个图就是加权图（Weighted Graph）。例如说中国公路网就是一个很好的“加权图”例子：每个城市是一个节点，每一条公路是一条弧。图中弧的权重对应于地图上的距离，或者是行车时间、过路费金额，等等。图论中很

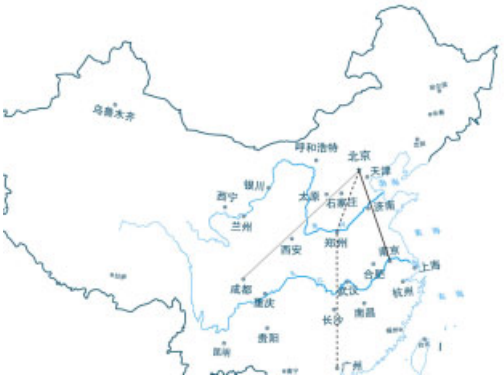


图2　中国公路图是一个特殊的加权图

常见的一个问题是要找一个图中给定两个点之间的最短路径（Shortest Path）。比如，想找到从北京到广州的最短行车路线或者最快行车路线。当然，最直接的笨办法是把所有可能的路线看一遍，然后找到最优的。这种办法只有在节点数是个位数的图中还行得通，当图的节点数（城市数目）有几十个时，计算的复杂度就已经让人甚至计算机难以接受了，因为所有可能路径的个数随着节点数的增长而呈指数（或者说几何级数）增长，也就是说，每增加一个城市，复杂度要大一倍。显然导航系统不会用这种笨办法——任何导航仪或者导航软件都能在几秒钟内就找到最佳行车路线。

所有的导航系统采用的都是动态规划的办法，这里面的Programming一词在数学上的含义是“规划”，不是计算机里的“编程”。动态规划的原理其实很简单。以上面的问题为例，当我们要找从北京到广州的最短路线时，先不妨倒过来想这个问题：假如已经找到了所要的最短路线（称为路线一），如果它经过郑州，那么从北京到郑州的这条子路线（比如是北京→ 保定→ 石家庄→ 郑州，暂定为子路线一），必然也是所有从北京到郑州的路线中最短的。否则，可以假定还存在从北京到郑州更短的路线（比如北京→ 济南→ 徐州→ 郑州，暂定为子路线二），那么只要用这第二条子路线代替第一条，就可以找到一条从北京到广州全程更短的路线（称为路线二），这就和我们讲的路线一是北京到广州最短的路线相矛盾。其矛盾的根源在于，假设的子路线二或者不存在，或者比子路线一还来得长。

在实际实现这个算法时，我们又正过来解决这个问题，也就是说，要想找到从北京到广州的最短路线，先要找到从北京到郑州的最短路线。当然，聪明的读者可能已经发现其中的一个“漏洞”，就是在还没有找到全程最短路线前，不能肯定它一定经过郑州。不过没有关系，只要在图上横切一刀，这一刀要保证将任何从北京到广州的路线一分为二，如图3。



图3　从北京到广州的路线必须经过图中粗线上的某个城市

那么从广州到北京的最短路径必须经过这一条线上的某个城市（乌鲁木齐、西宁、兰州、西安、郑州、济南）。我们可以先找到从北京出发到这条线上所有城市的最短路径，最后得到的全程最短路线一定包括这些局部最短路线中的一条，这样，就可以将一个“寻找全程最短路线”的问题，分解成一个个寻找局部最短路线的小问题。只要将这条横切线从北京向广州推移，直到广州为止，我们的全程最短路线就找到了。这便是动态规划的原理。采用动态规划可以大大降低最短路径的计算复杂度。

在上面的例子中，每加入一条横切线，线上平均有10个城市，从广州到北京最多经过15个城市，那么采用动态规划的计算量是10×10×15，而采用穷举路径的笨办法是10的15次方，前后差了万亿倍。

正确的数学模型可以将一个计算量看似很大的问题的计算复杂度大大降低。这便是数学的妙用。



“数学之美”系列文章原刊载于谷歌黑板报，得到高度评价。今年，作者吴军博士几乎把所有的文章都重写了一遍，把高深的原理讲得更加通俗易懂，让非专业读者也能领略数学的魅力。本文节选自《数学之美》，略有删改。特此感谢人民邮电出版社授权。

搜索引擎的查询意图识别

文 / 张帆

意图识别定义和难点

随着近年来互联网的快速发展，网络上的信息越来越丰富，而且信息的载体也越来越多样化，除文字之外，图片、视频、音乐等为载体的信息也越来越多，同时人们也越来越习惯于在互联网上搜索信息，而搜索引擎也就成为了人们搜索知识的最重要的手段。通常来说搜索引擎分为两个大类：通用搜索和垂直搜索。

- 通用搜索引擎：它会抓取收录互联网上一切有价值的页面，统一建立索引，以关键字匹配为基本的检索方式，以网页的标题、缩略内容和URL为展现形式。目前常见的通用搜索引擎有Google、百度、搜狗、搜搜等。
- 垂直搜索引擎：它以一特定类别为主题，只抓取和收录互联网上与主题相关的信息，根据主题本身的特点有针对性地建立相应的索引方式、检索方式、筛选方式，以及展现方式。而且由于垂直搜索的索引内容很多是人工添加的，所以准确度较高。目前常见的垂直引擎有一淘（购物）、去哪儿（机票）、安居客（房产）、智联招聘（招聘）、搜狗地图（地图）等。

通用搜索的优点是覆盖面非常广大，无论什么样的查询只要网上有相关信息，总是可以搜出来的，因此常常被用搜索信息的主入口，但由于其展示方式的局限性导致对于某些特定领域的查询不能得到很好地满足，而这些查询在其对应的垂直搜索中却有非常好的结果。意图识别的目的就是让我们通过用户的查询词来识别出用户的具体意图，并将查询词优化后输入到对应的垂直搜索引擎中，再将获取到的结果过滤优化后嵌入通用搜索引擎的结果中。

这样一来用户就不需要熟知所有的需求所对应的垂直搜索引擎，只要以通用搜索作为搜索入口就可以了，反映到具体的应用上比较出名的就是百度的阿拉丁和搜狗的全搜索。另外意图识别的应用范围也不只是传统的搜索引擎，对于一切非标准化或多模态输入的服务提供系统来说，都是必不可少的部分，例如汽车上使用的“语音导航系统”、现在越来越流行“个人数字助理”等。但与此同时，对于做好查询意图识别，我们也面临着一些困难。

- 查询输入不规范：用户在利用搜索引擎进行查询时所输入的查询往往是不规范的。有些用户会简单罗列关键词，通过查询词“北京 上海 车票”这种方式来检索北京到上海的火车票信息；还有些用户会使用复杂的人类自然语言，以“我要找下个礼拜北京到上海的便宜火车票”这样的形式来检索北京到上海的火车票信息；还有用户甚至会输入乱序的自然语言，比如“查找北京到上海的火车票，要下周的，要便宜的”。这就要求搜索引擎理解这些非规范的问题，并从其中解析出有价值的信息，并转换为垂直搜索引擎的输入方式。

■ 查询输入错误：用户通常是通过键盘来输入查询词的，因此常常会出现输入错误，但如果这种错误没有歧义的话，我们还是希望可以理解用户的意图，例如“北京道上海的火车票”，其中将“到”写成了“道”。这也就要求我们判断查询词是正常输入还是错误输入，如果是错误输入的话就需要纠正这些错误。

■ 查询词的多重意图：对于一个查询词可能会有多重意图，例如查询词“仙剑奇侠传”，就至少包含了游戏、电视剧、小说、音乐、图片5个大类，而且每一个大类又可以细分出更多小类，比如游戏大类就可以继续细分出游戏下载、游戏攻略、游戏购买等小类。而且对于这些多意图的查询词，我们不但要识别出它的意图集合，还需要知道各意图的强弱度，并根据这些信息来决定相应的展现方式。

■ 查询的时效性：意图识别是有时效性的，同一个查询词在不同的时间有着不同的需求。例如用户在iPhone 4手机的发布之前，搜索iPhone 4的主要意图是新闻意图，是为了了解iPhone 4的相关信息，而一旦发布之后，它的主要意图就变成购物了。这就要求我们需要根据时间来决定查询词的意图。

■ 垂直结果的可靠性：由于垂直搜索往往都是一些中小型网站，而且基数较为庞大，所以它们的搜索质量很难有保障，这就需要我们在采集数据后进行数据的后处理，来保证其内容的相关性。

意图识别实现方法

意图识别系统中比较通用的方法有三大类：基于查询词词表的方法、基于查询词解析和规则的方法，以及基于查询词统计模型的方法。它们基本可以满足绝大多数的意图识别，而且它们不是相互独立的方法，需要整合在一起来解决问题。

基于查询词词表的方法

这是最简单也最直接的意图识别方法，通过查询词的白名单的精确匹配、模糊匹配来进行意图识别，还可以通过查询词的预处理来提高召回和减少白名单的索引量。对于白名单的建立，通常我们会从已有的查询词日志和点击日志中的来挖掘出与特定意图想对应的候选白名单，之后由人工过滤这些词表，并总结出其中一些基本规则，再整合一些已有的知识性的词表来生成更多白名单词表。对于查询词预处理，通常会去掉无意义的停用词，对同类型的词进行归一化，词间顺序的归一化等方法，具体的方法会根据不同的类别而作调整。

这种方法的优点是简单易实现，能很好地解决高频查询词的意图识别，而且准确率非常高，但同时它也存在召回较差、人工比例较高的问题，而且无法很好地解决长尾查询。因此它比较适合那些查询词非常集中的类别，例如天气预报、电视台节目、节假日、百科等类别。

意图识别系统中比较通用的方法有三大类：基于查询词词表的方法，基于查询词解析和规则的方法，以及基于查询词统计模型的方法。

基于查询词解析和规则的方法

这是一种以查询词解析为基础的意图识别方法，通过对查询词的内容解析和领域词典匹配来进行意图识别，此外还可以再利用模糊的黑白名单来进一步提高精度和召回。查询词解析就是通过知识类词典或者正则表达式的方法来解析查询词，并且从中提取出日期、时间、数字等规则性较强的信息并将其标准化。领域词典匹配则是根据类别查询日志挖掘出的类别中的词汇，来判断查询词的意图。

这种方法的优点是意图识别的精度和召回都非常高，而且可以规则化查询词中不规则的有价值的信息，但同时它的适用面也比较窄，只适用于一些查询比较分散但规则性很强的类别，例如汇率换算、度量衡换算、计算器等类别。

基于查询词统计模型的方法

基于统计模型是意图识别最常用、最有效的方法，不但效果好、召回高，而且可以在持续获得训练数据的情况下完全自动更新模型，也就解决了时效性

的问题，但同时它也存在着实现较为复杂、训练数据获取困难、训练数据更新困难等问题。

■ 基于查询词统计模型：以查询词为基础的分类模型，通过查询词的正例和负例的训练语料判别查询词是否具有某一意图的二分类统计模型，对于有包含关系的类别还可以建立多层次的分类模型，例如“仙剑奇侠传攻略下载”，我们由游戏意图的分类模型判断出是有游戏类意图的，再通过二级分类模型在游戏意图的基础上继续分类到游戏攻略的意图上。

基于统计模型是意图识别最常用的最有效的方法，不但效果好，召回高，而且在持续获得训练数据的情况下可以完全自动更新模型。

训练语料是统计方法解决问题的根本，它的数量和质量直接影响到我们后面训练模型的效果，因此这部分非常重要。我们有通常三种方法来获取查询词训练语料：人工标注、垂直搜索查询日志和点击日志挖掘。第一，最简单的方法就是人工标注，但这样成本很高，而且无法保证标注语料的质量，特别是当多人协作标注时，标准的不统一将很难避免，而且也难以得到持续的更新。第二，通过垂直搜索引擎获取查询日志也是我们可以利用的方法，这种方法既可以保证查询词的精度和标准的统一，也可以解决自动更新的问题。但各家垂直搜索都视搜索日志为自己的核心机密，并不会共享出来，除非有自己的垂直搜索引擎，不然很难持续拿到日志数据，所以也比较难以实现。第三，我们可以通过自身日志挖掘来获取训练语料，根据用户搜索后的点击行为来判断查询词的查询意图，这种方法相对容易实现，对于点击行为较为丰富的查询词可以得到很高的精确度，而且可以设定更新流程来每日自动生成查询词，也可以很好地解决更新问题，但缺点也比较明显，收集到的数据并非随机分布的，而是比较偏向于高频词，可能会影响最终模型的效果。

除了查询词以外，我们也可以考虑使用一些其他的数据来模拟查询词，来训练单独的模型辅助查询词模型，例如地图类的意图识别可以使用POI词表、地址表等信息来建立辅助模型，再比如购物类的意图识别可以使用商品的名称等信息来建立辅助模型。这些模型的精度会低于查询词统计模型，但如果使用得当，可以对召回有很好的补充。

另外我们也可以通过对查询词的预处理来降低数据稀疏问题，可以使我们用更少的数据获得更密集、更精确的模型，我们可以配合规则或者其他实体识别模型，来归一化一些已知的实体，例如“下载搜狗输入法3.5”中我们先将查询词转化为“下载[软件名][版本号]”再进行训练和解码，这样只要一个三元的特征就可以搞定了，不然可能需要|软件数|*|版本号数|个三元模型才能解决，且训练语料很难充分到包含所有可能性，而通过实体归一的方法，可以很好地解决这种问题。

■ 基于查询词扩展统计模型：由于查询词本身为短文本，所以分类难度较大，而且只是根据字面意思区分，而并不知道真实世界中的知识，会有一些无法区分的情况，例如查询词“家常菜”根据查询词分类模型计算会认为是烹饪类的需求，但实际上它是一个当前正火热上映的连续剧。在这种情况下，使用查询词本身分类是无法解决的，所以就需要使用的查询词扩展分类来协助解决。最常用的查询词扩展方法就是获取查询词的搜索结果，再根据搜索结果来分类，另外还有使用百科类网站来做查询的，但并不是常用方法，在此就不做深入讨论了。

基于搜索结果扩展的分类模型就是要通过搜索结果中各种意图的结果的条目数、比例和位置，综合判断出查询词所包含的意图和意图的强度，而对于每条结果的判别是基于结果的标题、摘要、网站主题、URL信息来综合判断，对于标题和摘要需要建立对应的文本模型，对于网站主题则需要利用主题模型来判断网站的类别或者二级域名的类别，对于URL则需要挖掘常出现的后缀信息的意义。最后再根据综合判断来计算出整体的意图集合和强度集合。

我们还可以根据垂直搜索的结果来进一步印证查询的意图，对于识别出意图的查询词带入到对应

的最好的垂直搜索引擎中获取相应的查询结果来计算结果的数目、匹配程度等信息，从数据端再次验证识别的结果，从而获得更高的准确度。

上述的几种方法并非独立存在的，单独使用都不会得到非常理想的效果，合理整合在一起才能发挥最好的效果。通常的组合是由基于查询词词表的方法解决高查询量的词，用基于查询词的统计模型解决长尾问题，再由查询扩展统计模型做最终验证，最后再由规则模型解析出其中的重要信息，填写到对应的一家或几家垂直搜索引擎的入口中获取查询结果，综合过滤和排序再展现给用户。

无类别概念的意图识别

目前的意图识别基本都是基于类别的，先是通过找出查询词与类别的关系，这样可以大大简化目前的识别工作，使自动化程度提高，而且垂直网站的覆盖度和查询词的召回会得到极大提高，不过在这之前，我们还是需要解决一些问题，例如自适应的网站结果解析方法，以及更加合理的展现方式等。

基于上下文分析的个性化意图识别

目前的意图识别通常与上下文无关，也就是说对于一条查询词无论谁查，什么时候查，所得到的意图识别结果都是相同的，但显然同一个查询词对不同的人来说意图是不一样的，因此可以根据长期个性化和短期个性化两个方面来优化这个问题，更加精确地命中用户的查询需求，长期个性化就是我们根据用户的长期搜索行为和浏览行为来记录来用户的长期偏好，短期个性化就是根据用户短期内的查询上下文和浏览上下文来判断用户的短期兴趣点，并以此作为依据来优化意图识别。

精准意图理解

目前的意图识别更多的是较为粗放的满足用户的意图，只是根据统计模型来判断用户输入的查询词的意图，再直接取回垂直的搜索结果或者预设的结构化数据来展现，并没有精确的理解查询词中的每一个语义，如果可以更精确地理解查询词的含义，并做针对性的筛选展示，可以更好地满足用户的需求。

语音应用的意图识别

随着语音技术的发展和苹果公司的Siri智能电子助理的推出，大家也越来越重视基于语音技术的应用或服务，而语音恰恰是基于自然语言形式的非规范的输入，甚至还包括图片、语音、GPS等信息的多模态输入，这样意图识别也就成为必不可少的组成部分，但针对语音的输入还有很多的新的问题需要解决。

意图识别的展望

随着目前信息载体的多样化发展和输入方式的多模态化、智能化发展，意图识别将会更加重要，在通用搜索引擎方面深度挖掘暗网信息、个性化意图识别、精准化意图识别都是未来需要继续深入研究的方向。另外，随着语音技术的发展和智能终端的发展，也会有更多结合语音功能的智能服务系统出现，我们也会需要这对这些系统的特点来设计基于它们的意图识别技术。



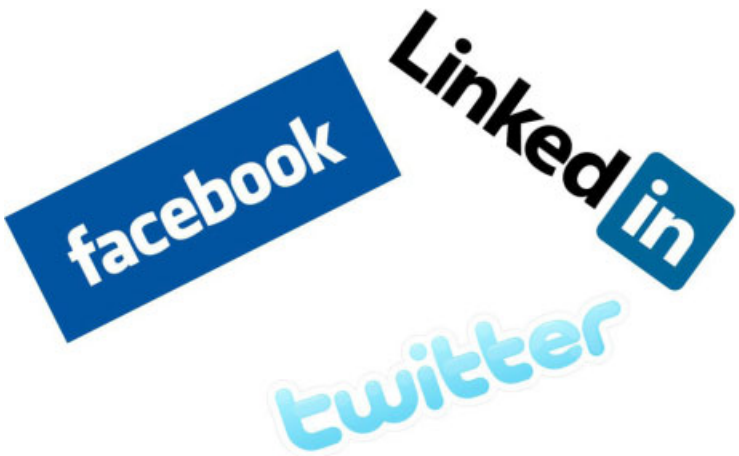
张帆

就职于搜狗网页搜索研究部，从事查询意图识别相关的工作，法国巴黎大学自然语言处理硕士，法国梅斯大学模型优化硕士，主要研究方向为自然语言处理、搜索引擎、智能对话系统和机器翻译。

社交网络：数据科学家眼中的金矿

文 / 张文浩

随着Facebook的上市，社交网络再次成为人们关注的焦点。与传统的论坛、博客相比，社交网络是虚拟世界与现实世界的桥梁，在互联网上将现实生活中人与人之间的关系建立起来。从社交网络的分类来看，Facebook、Twitter、LinkedIn分别代表三种不同的社交网络。Facebook是基于朋友之间强关系的社交网络，有助于朋友之间关系的维系和改善；Twitter是基于单向关注的弱关系的社交网络，这样的网络有利于塑造意见领袖和消息的传播；LinkedIn是面向商务人士的职业社交网络，帮助用户利用社交关系进行商务交流以及求职招聘。三种社交网络每天都会产生大量的用户数据（UGC，User Generated Content），并且具有空前的规模性和群体性，吸引着无数研究者从无序的数据中发掘有价值的信息。这就像概率统计中经常举的投硬币算其正反面概率的例子，从几次的投掷结果中很难看到规律，但通过几万次的大量投掷实验，便很容易看出正反面的出现次数几乎相等的规律。社交网络上产生了大量的规模化、群体化的数据，吸引了包括计算机科学、心理学、社会学、新闻传播学等领域专家和学者对其进行研究和探索，希望能够借助更强的社交网络的分析和处理能力发现更多人类尚未探索出的规律。



对于社交网络的分析和研究范围很广，也存在着许多有意思的研究课题。例如，在社交网络中社区圈子的识别（Community Detection）、社交网络中人物影响力的计算、信息在社交网络上的传播模型、虚假信息和机器人账号的识别、基于社交网络信息对股市、大选以及传染病的预测等。社交网络的分析和研究是一个交叉领域的学科，所以在研究过程中，我们通常会利用社会学、心理学甚至是医学上的基本结论和原理作为指导，通过人工智能领域中使用的机器学习、图论等算法对社交网络中的行为和未来的趋势进行模拟和预测。

社交圈子的识别

与一般的以内容为导向的论坛等社区不同，社交网络最核心的就是人与人的关系，以及所形成的社交圈子（社区），然而每个人根据自己的关系不同及兴趣不同可以属于多个社交圈子。在社交网络中我们发布的所有信息流，都是通过我们的关系圈，逐层向外传播的；我们收到的消息也直接来自我们所关心的人，更外围的消息也必须逐层传播才能接触到终端用户。因此，如何发现社交圈子是社会关系网络分析中一个很重要的基础性的研究。社交圈子示例如图1所示。

利用计算机来处理社交网络往往会将整个社交网络看作是一个图的结构，每个用户就是图中的节点，人与人之间的关系就是节点之间的边，根据不同类型的社交网络，所构成的图可以是有向图也可以是无向图，关系的强弱也可以利用边上

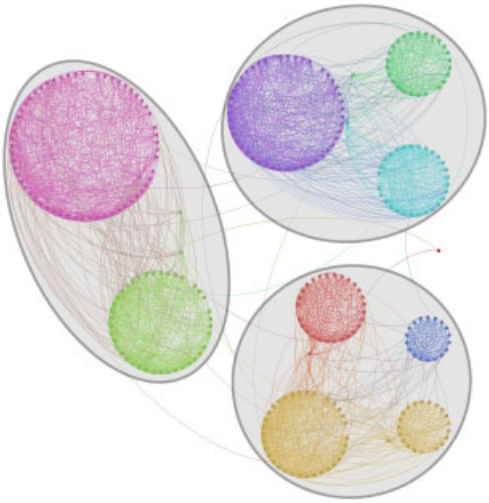


图1　基于OSLOM算法的社区发现效果图（来自www.oslom.org）

不同的权重来体现。对于社交圈子的发现算法来说，社交圈子的质量依赖于圈子内成员的关系的紧致度以及不同圈子间的分离度。但对于数以亿计的节点来说，目前的圈子发现算法还很难处理特大规模的数据，因此很多研究者提出了启发式的方法去减少程序处理的复杂性，对最终结果进行近似的求解。

然而实际的社交圈子是一个更为复杂的网络，因为用户会具有多种兴趣，可以属于多个社交圈，发现这种圈子的研究也被称为重叠社区的发现。一种比较简单的启发式方法是，以网络中度很大的节点作为初始的圈子，然后把对圈子贡献最大的邻接节点依次加入到圈子中，直到全局贡献度达到极值，并形成一个圈子。如果存在对多个圈子贡献度都很大的边界节点，则将其加入到多个圈子中。近期也有人提出了使用标签传播（Label Propagation）算法以及粒子群算法来解决重叠社区的发现算法。

社交圈子发现算法并不仅局限在用户主动建立起的关系上，其更重要的价值在于对用户非显性的潜在关系发现。从社交圈子发现的结果中，我们能够更加清楚地看出属于一个圈子的人群。当然，社交圈子也有多种划分方式，例如关系型社交圈子、兴趣型社交圈子等。在算法中以亲密度为首要指标和以兴趣为首要指标，也会得到不同的社交圈子划分。

由此引申出的一个问题是，线上的圈子与线下的真实社交圈子是否是一致的？当两个人在社交网络中互动很频繁时，他们在线下是否也是真实的好友？从算法的角度来说，这是个很难解决的问题，但如果我们换一个角度来思考这个问题，想想我们的线下联系方式，如果A跟B互相拥有对方的手机号，那他们是线下真实好友的可能性就非常大了。包括飞信、米聊、微信等产品，如果真的能够做成基于手机通讯录的社交网络，我们就可以通过异构的社交网络对社交圈子进行综合性的判断，其价值不可估量。

影响力的计算

在社交网络中，意见领袖因为其在网络上强大的影响力会对信息的传播，以及普通用户的行为造成巨大的影响。以新浪微博为例，最直观的影响力表现之一就是加V认证的名人，发一条正在吃饭的微博，也能得到数百次的转发，然而对于一般用户，一条微博的转发次数能上两位数，便足以为之欢呼雀跃了。

因此，与现实社会一样，社交网络中的人也存在不同的阶级和不同的影响力。然而影响力应该如何来衡量和计算呢？我们前面已经提到，计算机在处理社交网络时，往往使用图的结构，这与搜索引擎中的结构相一致，如表1所示。因为在搜索引擎中，图的节点是网页，边是链接，然而搜索引擎中的PageRank算法是对网页进行排序的算法。如果我们将PageRank使用在社交网络上，就可以对人的影响力进行迭代的计算了。除了PageRank算法外，还有W-entropy等算法也被应用在社交网络的影响力计算当中。

表1 社交网络与搜索引擎对图结构的不同定义

	搜索引擎	社交网络
节点	网页	人
边	网页链接	关系

然而对于每个人来说，其在不同领域的影响力也是不一样的。例如，李开复的影响力主要在科技领域，黄健翔的影响力在体育领域，薛蛮子的影响力主要在投资和公益的领域。因此如何评价一个人在不同领域的影响力也是一个很重要的问题，有学者提出了基于主题级别（Topic

Level）的影响力评价模型TAP（Topic Affinity Propagation）来尝试解决这个问题，该算法应用在大规模社交网络数据中显现出了较好的效果。

在国外，Famecount、Klout等公司设计了算法对社交网络中每一个人的影响力进行打分。有些公司甚至出现了根据个人网络影响力的不同，在现实生活中提供差异化服务的模式，例如，香港国泰航空对Klout打分不低于40分的用户，可以享受机场贵宾休息室。虽然该行为也遭到不少人质疑，认为这是“势利”的做法，不过也可以将其看作是对网络影响力应用在商业模式上的一种新型探索。在国内，新浪的微数据和Miu+也在微博的影响力计算中做出了一些探索，目前国内在该领域还有较大的发展空间。

信息传播的建模

在社交网络上，每个人都是个自媒体。与传统媒体依靠内容作为传播主题的形式有所不同，社交网络上的信息传播，更加依赖于发布者的影响力以及社会关系，通过好友或粉丝的关系将信息扩散到社交网络中。这种信息在社交网络中会被好友及粉丝看到，并以一定的概率被分享和转发，从而进行传播。图2对一条微博的传播过程进行了可视化的显示。

部分学者用传染病在人群中的传播、谣言在社会中的传播等现象来类比并刻画社交网络中信息的传播，进而利用传染病动力学及复杂网络理论来对社交网络的传播行为进行建模和预测。更直观地说，如果将整个社交网络看作一个图的结构，把社交网络中的用户看成图中的节点，并将用户之间的关系看为图的边，那么信息的传播的过程就是从起点用户的节点开始，沿着相邻边进行信息的传播，相邻的节点用户会根据时间及主题不同，会以一定概率传播或者终止该信息。对于传染病动力学模型来说，通常将网络中的节点定义为三类：传播节点、未感染节点和免疫节点。传播节点的特点是接受并有能力传播邻居节点信息；未感染节点没有接受过来自邻居节点的信息，但有机会接受信息，即有概率会被感染；免疫节点表示该节点已接受邻居节点的信息，但不具有传播能力，由此可以定义一些传播规则：

- 如果一个传播节点与一个未感染节点接触，则未感染节点会以概率成为传播节点；
- 如果一个传播节点与一个免疫节点接触，则传播节点会以概率成为免疫节点；
- 传播节点不会无休止地传播，会以一定速度停止传播，变为免疫节点，无需与其他节点接触。

由此便可以通过传染病动力学中的方法对信息的传播过程建立状态转移方程。在确立传播模型后，我们可以通过检验起点传播者节点的度（也就是好友或粉丝的数量）、关系强弱（边的权值），对信息传播造成的影响程度，从而发现信息在社交网络传播中的规律。



图2　单条微博信息的传播图（来自www.doodod.com）

虚假用户的识别

虚假信息和虚假用户的识别在社交网络的深入研究及实际应用中是一项基础性的工作，并具有重要的意义。信息在社交网络上的传播过程中，不免会遇到虚假内容或虚假的水军用户进行干扰。如果能够识别出虚假用户和虚假内容可以更好地还原出舆论的真实想法和状态，为企业的市场营销以及政府了解民意提供更加真实有效的数据。一般来说，识别社交网络的虚假用户要比匿名论坛相对容易，因为可以从更多的维度进行考察。在社交网络上，虚假账号一般不会有真实的社交互动，关系网络中大部分链接也都是虚假账号，此外通过账号的转发行为及内容的识别判断来有

效地辨别虚假用户。我们在新浪微博上对虚假用户的判断采用了以下8种用户行为特征。

- 博主的创建时间的一致性
- 博主的头像和名字
- 关注与粉丝比例
- 博主的粉丝质量
- 发布微博数量
- 最近200次转发的对象分布
- 转发同一条微博的频率
- 转发时所写的内容

针对以上8种特征，利用机器学习的分类算法训练模型，并利用模型进行后续虚假用户的预测，可以有效地发现虚假用户，在舆情分析中将其剔除，还原出真实的信息传播情况及舆情（公众发表在网络上的言论）状态。

用数据预测未来

社交网络数据最吸引人的研究工作就是对未来的预测。社交网络每天吸引数亿人在网络上发布自己的数据、状态、心情，这种规模化并具有群体性的海量数据给了数据科学家从海量数据中发现人类未知规律的机会。

美国科学家通过监控Twitter中公众的情绪数据，发现公众的情绪数据与很多社会现象及事件具有很强的相关性。例如，有些研究者发现无论是“希望”的正面情绪，还是“害怕”的负面情绪的体现都预示着美国股市指数的下跌。有研究者认为，只要有公众在社交网络的情绪突然改变，都会反映出对股市的不确定性，因此可以利用这种信号来预测股市未来的走向。

在流行病预测方面，英国的科学家根据Twitter的数据来跟踪流感的爆发。他们主要基于用户发布信息中的关键词，例如“我头痛（I am having a headache）”等，并结合用户的发布地点，按区域与英国卫生部的官方数据进行了比较，最终建立起一个预测模型。创业团队“SickWeather”甚至以预测疾病为主题开展了自己的创业项目。

还有很多研究者也利用数据挖掘的方法对电影票房、美国大选的趋势和结果进行预测，并取得了令人惊喜的成果。

然而，我们对于利用社交网络数据的预测能力的态度也不能过于乐观，因为社交网络的预测是基于海量数据的，但目前对于海量文本数据的分析算法尚未达到理想的准确率。尤其对于从文本信息来进行情绪判断这个看似简单的问题，其本质是自然语言处理与情绪心理学的交叉问题。但目前的自然语言处理方法主要利用概率统计的方法，以及词法和句法的分析进行解读。对文本情绪的判断也以基于词库及语法结构的判断和基于机器学习的方法为主。然而这些方法对于稍显复杂的、尤其是带有反讽和隐含意的语言很难进行有效的判断。此外，对于社交网络的使用群体不能完全代表有效的人群，因为使用社交网络的人群与年龄、地域、种族等方面都有很大差异，因此仅利用社交网络产生的数据进行预测很可能会与最终结果产生偏差，所以从人群角度进行科学有效的取样方法对于社交网络预测也是尤为重要的一个环节。

总结

人们对社交网络数据的认识和挖掘还处于相对初级的阶段，对这种大规模、高维度数据的挖掘方法还在不断地演化。目前来看，文本语言的情感分析、社交网络的传播预测等很多基础性问题还不能得到有效解决，对深入研究社交网络造成了一些限制。但随着人工智能研究水平的不断提高，尤其是认知神经科学与人工智能技术相结合的研究，让我们看到了人工智能的新希望。当我们真正有能力解决这些问题以后，社交网络将会成为帮助我们预测未来趋势的有利工具。然而，充分使用社交网络数据也意味着暴露用户越来越多的隐私，因此，如何能够在用户隐私和数据完整中找到一个平衡点，也是今后数据工作者所要面临的问题。



张文浩

独到科技（Doodod）创始人，清华大学计算机系博士，主要关注社会关系网络分析、文本情感挖掘等领域的研究和发展。

标签传播算法在微博用户兴趣图谱的应用

文 / 张俊林

“兴趣图谱”是近几年在互联网产业界被广泛提及的个性化概念，所谓“兴趣图谱”，就是根据用户的行为以及用户产生的内容等方面的数据来从中得出用户可能的兴趣点。在获知用户兴趣图谱的基础上，可以为用户量身定做或者推荐用户感兴趣的潜在信息，例如可能感兴趣的其他相似用户，以及可能感兴趣的内容等，也可以据此向用户推送广告信息。正是因为兴趣图谱如此重要，所以包括Facebook、Google、Twitter、新浪等大型互联网公司以及众多初创公司正致力于更加准确地从数据中推断用户可能的兴趣点。

新浪微博是最近几年国内非常流行的社交网站，微博用户可以在微博平台上分享观点、传播有益或者有趣内容、与感兴趣的人交流互动等。微博用户作为独立的行为主体，产生了大量个性化的信息，而根据这些信息可以很有效地对用户可能的兴趣点进行挖掘，来获知用户的潜在兴趣所在，构建完善的用户兴趣图谱。在此基础上，还可以为用户的订阅信息流按照兴趣排序，筛选高质量内容，有效降低用户获取有用信息的成本，其重要性不言而喻。

关于构建用户兴趣图谱的资料很多，本文主要介绍其中比较重要的基于文本内容的信息：用户标签，以及从用户发表微博信息中自动提取出的用户兴趣词。通过利用微博用户自己产生的内容并结合用户社交关系，可以有效提升基于内容的用户兴趣图谱挖掘的准确性。

标签传播算法

标签传播算法是在社交网络分析领域中被广泛采用的算法，经常使用这个算法来自动挖掘社交关系中的“团结构”（通俗的讲就是不同的圈子）。之所以这个算法能够被广泛使用，主要因为它有两个明显优点：首先，这个算法思路非常直观和简洁，易理解、易实现；其次，这个算法的机制也非常容易实现对大规模数据进行处理，具备很强的实用性。

我们通过描述如何利用标签传播算法来自动发现社交关系中的“团结构”来大致讲解其算法思路。假设存在一个社交网络S，其包含若干节点，节点之间由边相互连接。这是足够抽象的一个图结构，很多具体应用可以映射为这种抽象图结构，例如在微博环境下，每个节点可以代表一个用户，而如果节点之间有边连接，则代表其有关注关系或者是有互动行为，节点和边的所代表的含义与具体应用密切相关。

对于社交网络S，如何通过标签传播算法自动发现其中的密集连通子图（即“团结构”）？算法的初始阶段，可以为图中每个节点赋予一个独一无

二的标签L，这个标签即为节点的初始标签值。接下来通过多轮迭代计算，通过社交关系（即图的边）将标签向其他节点传播。在一轮迭代运算过程中，某个节点将根据与其有边联系的其他节点的标签来决定自己此轮应该赋予哪个标签。基本原则是：将其邻居节点的标签中出现次数最多的那个标签赋予自己，如果邻居节点的标签数目一样多，无法找出最多个数标签，则随机赋予一个标签。每个节点都根据这个原则重新确定本轮中本节点应该获得的新标签，这样就完成了一轮标签赋值运算。当进行了多轮运算后，绝大多数节点的标签不再发生变化，这就是最终计算结果。如果图中节点拥有相同的标签，则可以被认为属于同一个“团结构”。之所以能够使用这个算法发现图中的密集连通子图，是因为密集连通子图的节点通过几轮迭代能够很快获得相同的标签，而处于两个不同密集连通子图边缘的节点，其标签则取决于其和哪个子图的关系更密切。

图1所示是一个由5个节点连接起来的密集连通图。刚开始每个节点都赋予不同的标签，即a、b、c、d、e（第一个子图）。算法进行过程中，标签为c的节点根据邻居的标签修正自身标签，发现其他4个节点标签各不相同，不存在占优势的标签，于是随机取值修正为标签a；接下来标签为e的节点发现邻居中最多的标签是a，于是修正自身标签为a，如此每个节点不断修正，最后所有节点的标签都会标为a。通过这种标签传播方式，密集连通子图中的节点标签都会标为同一标签，在社交网络“团结构”发现任务中，也就是说这些节点属于同一个社区。接下来的内容我们介绍如何使用标签传播算法来帮助发现用户兴趣图谱。

使用标签传播算法计算微博用户兴趣图谱

构建用户兴趣图谱可利用的信息来源形形色色，例如用户发布微博的内容以及其所属类别、用户

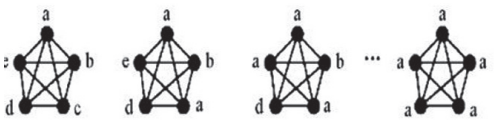


图1　标签传播算法示例

自标签、用户社交关系及社交行为、用户参与的群组等，新浪微博内部已采取了多种技术手段利用多种信息来源建立了比较准确的用户兴趣模型。下面主要从文本信息的利用角度介绍如何使用标签传播算法探索用户兴趣模型，文本内容主要包含两类信息：一类是从用户发布的微博中挖掘出的用户兴趣词；另一类是用户填写的标签信息。标签传播算法可以将社交因素引入文本内容的挖掘，使得两者有很好的结合，更好地挖掘用户潜在的兴趣。尤其是对于发布较少内容且不标注自标签的不活跃用户，通过社交行为的引入能够有效拓展对其兴趣的发现可能性。

基本原理

标签传播算法是一个标准的图挖掘算法，即其计算对象是由节点和节点之间的边构成的图。对于微博用户兴趣词挖掘这种应用场景而言，图中的节点就是每个微博用户，以微博用户ID代表节点唯一编号，而节点之间的边则代表用户之间的互动关系，即如果用户A和用户B在微博平台有互动关系，则可建立一条AB节点之间的边。所谓“互动”，包括用户之间的转发、评论、@提醒，以及是否互粉等交互行为。很明显，如果两个用户之间的互动越频繁，那么两者之间的社交关系越紧密，而亲密的社交关系往往蕴含着潜在的兴趣关联或者较强的线下社交关系。

通过微博用户的社交行为，就可以建立一张数亿节点构成的大图，而对于标签传播算法来说，每个节点还需要标注初始的“标签”信息，在此种应用环境下，每个节点的标签就是用户填写的自标签以及从用户所发布的微博中提取的初始用户兴趣词列表，如果用户没有发布过微博或者尚未填写自标签则可以将该用户的标签信息置为空集合。这样就完成了标签传播算法所需的初始数据及图结构。这里与常见的标签传播算法不同的一点是：每个图节点包含的标签并非一个，而是根据权值确定的K个权值最高的标签或者兴趣词作为节点的标签集合。

之后即可按照标签传播算法的逻辑来通过多轮迭代完成标签的传播和节点标签集合的更新过程，迭代次数可以事先指定或者节点标签集合变化不

大的时候终止算法流程。在每一轮迭代过程中，图中每个节点都依次执行以下步骤来更新其标签集合。

1. 找到与节点A有边联系的邻居节点，形成邻居节点集合S，S中的这些节点代表了与用户A有过互动行为的用户集合，同时把A也放入集合S中；

2. 统计集合S所有用户中出现过的不同标签的频次，比如一个标签在S中的10个节点的标签集合中出现过，则计数为10。用户A本身初始的标签按照频次f参与计数，即给用户本身的标签一个较大的初始频次；

3. 将第二步获得的标签按照其频次和标签的IDF值使用类似于IFIDF计算框架的方式计算其权重，并按照权重得分高低排序，取Top K标签作为节点A的新标签集合。

通过以上方式，就完成了某个节点的标签集合更新，这里需要说明，第3步中为何要引入IDF值，而不是直接利用标签频次来进行排序。考虑到用户标签中包含一些非常通用的内容，比如“音乐”、“80后”等，这些标签过于常见，相当比例微博用户会带有这种标签，而因其过于普遍所以对于刻画用户兴趣来说意义不大，故引入IDF因子对这类标签进行降权。

单词或者标签的IDF因子是衡量其相对重要性的一种计量标准，这个因子广泛使用在搜索排序或者文本处理领域中。同时，在第三步还需要增加调节因子来控制频次和IDF两者的重要性，这样就可以对算法进行灵活调整。此外，为了避免过于普通标签的影响，还可以对标签集合进行过滤，将IDF值过高或者过低的标签过滤掉，只保留信息含量最高的中间段标签。

图中每个节点都如上述方式更新其标签集合，经过若干轮的迭代计算，即可获得用户最终的兴趣标签。

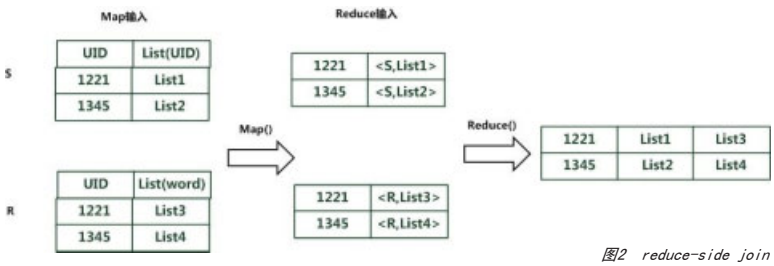


图2 reduce-side join

标签传播算法应用在捕获用户兴趣词领域，其基本思想是不仅仅利用用户自身产生的内容（自标签以及发布微博内容），还利用用户的社交关系，将用户所属社交圈子的共性信息作为用户兴趣的有益补充，更加准确地刻画用户兴趣图谱。尤其对那些发布内容较少的用户，其补充作用价值更加明显。

大规模数据计算

由于近几年的快速发展与普及，目前微博用户数以亿计，在如此大规模的用户数据情形下，如何快速计算用户的兴趣词就成为实际需要面对的工程问题。为了能够处理如此大规模的数据，采用并行分布式计算平台就成为必需。我们采用了Hadoop平台，将上述算法改造为若干组序列相连的MapReduce任务来完成实际工程计算。

数据聚合处理

由于后续计算所需原始数据有两个不同来源，一个来源是用户社交数据，以<uid,list(uid)>的形式存在，代表与uid有互动行为的其他用户列表；另外一个是用户的初始标签信息，以<uid,list(word)>的形式存在，代表uid对应的初始标签集合。为了方便后续算法正常运行，首先需要对这两类数据进行处理，来将某个用户的两类信息融合在一起，即形成<uid,list(uid),list(word)>的数据形式。

这个数据聚合处理步骤是典型的join操作，因此我们采取标准的Reduce-side join的方法来对其进行处理，即在Map阶段将数据来源（R和S）追加到对应的value字段中，在Reduce阶段根据相同uid的不同来源字段重新构建其对应的value，以完成join操作。其对应的计算过程如图2所示。

通过以上方式可以很方便地完成大规模数据的不用数据来源的join任务。

标签传播算法计算

当每个用户的数据经过聚合处理已经组织成为<uid,list(uid),list(word)>的形式后，则可以使用MapReduce任务来进行后续的标签传播算法计算。其计算逻辑如下列伪码所示：

```
map(String key, String value):
// key:   uid
// value: list(uid)+list(word)

/*解析value*/
list uidList=parse(value)
list wordList=parse(value)

/*标签传播*/
for each u in uidList:
EmitIntermediate(u, AsString(wordList));
If u==key: //如果是节点自身，传播f次
   int i=0
   while i<f:
      EmitIntermediate(u, AsString(wordList));
      i+=1

reduce(String key, Iterator values):
// key:    uid
// values: list(word)列表

/*统计标签频次*/
map<string,float> tfCounter;
for each v in values:
   list wordList=parse(v)
   for each w in wordList:
        tfCounter[w]+=1

/*计算标签权值*/
For each label in tfCounter.keys():
    float weight=TfIdf(label)
    tfCounter[label]=weight

/*取Top K*/
List topLabelList=GetTopK(tfCounter,K)

Emit(AsString(topLabelList));
```

在Map操作阶段，key为某微博用户的uid，value为该用户的社交关系列表及其标签列表。Map函数从value中分别解析出两种数据，并将自身的标签列表传播给其邻居节点，同时为了增强自身原有标签的权重，会给自己传播f份标签数据。

在Reduce阶段，key为某微博用户uid，values为与uid有边相连的其他微博用户传播过来的标签列表序列。Reduce函数统计不同标签的频次，并与标签的IDF值拟合出标签权值，之后取得分最高的K个标签作为该uid更新后的标签集合。

通过“数据聚合处理”和“标签传播算法计算”两个序列MapReduce任务，即可完成一轮用户标签集合的更新过程，用户可以指定迭代次数，就可完成整个计算流程，形成最终的计算结果。

效果示例

通过使用标签传播算法，可以充分利用用户本身的标签数据，同时通过社交关系发现用户的群组兴趣，这样可以更好地对其潜在兴趣进行有效挖掘。我使用上面描述的方法获得了部分用户的Top 10标签，结果如表1所示，可以看到，还是比较准确的。

表1　利用标签传播算法获取新浪微博用户标签

微博用户名	Top 10 标签
张俊林say	自然语言处理/推荐系统/数据挖掘/机器学习/Google/云计算/NoSQL/信息检索/计算语言学/搜索引擎
刘江CE	云计算/程序员/android/社会化媒体/产品设计/架构师/CSDN/研发管理/移动开发/CTO
梁斌Penny	社交网络/机器学习/数据挖掘/自然语言处理/云计算/搜索引擎/数据分析/Python/信息检索/分布式计算
互联网的那点事	互联网/用户体验/新媒体/创业/天使投资/B2C/UED/创业者/交互设计/产品经理
创业最前线	创业/天使投资/VC/创业家/商业模式/企业家/风险投资/PE/领导力/创业者
李开复	创新工场/知乎/社交网络/移动互联网/天使投资/lbs/创业家/智能手机/豌豆荚/VC
周鸿祎	风险投资/奇虎/浏览器/互联网/360/杀毒/产品经理/创业/iPad/社交游戏
雷军	移动互联网/安卓/智能手机/米聊/天使投资/小米手机/iPhone/3G/创业/MIUI



张俊林

《这就是搜索引擎：核心技术详解》作者，新浪微博研发人员，中科院软件所博士，主要研究方向为自然语言处理、搜索技术、推荐系统及机器学习。

基于大规模语料的新词发现算法

文 / 顾森

对中文资料进行自然语言处理时，我们会遇到很多其他语言不会有的困难，例如分词——汉语的词与词之间没有空格，那计算机怎么才知道“已结婚的和尚未结婚的”究竟是“已／结婚／的／和／尚未／结婚／的”，还是“已／结婚／的／和尚／未／结婚／的”呢？

这就是所谓的分词歧义难题。不过，现在很多语言模型都已能比较漂亮地解决这一问题了。但在中文分词领域里，还有一个比分词歧义更令人头疼的东西——未登录词。中文没有首字母大写，专名号也被取消了，这叫计算机如何辨认人名地名之类的东西？更惨的则是机构名、品牌名、专业名词、缩略语、网络新词等，它们的产生机制似乎完全无规律可寻。最近十年来，中文分词领域都在集中攻克这一难关，自动发现新词成为关键的环节。

挖掘新词的传统方法是，先对文本进行分词，然后猜测未能成功匹配的剩余片段就是新词。这似乎陷入了一个怪圈：分词的准确性本身就依赖于词库的完整性，如果词库中根本没有新词，我们又怎能信任分词结果呢？此时，一种大胆的想法是，首先不依赖于任何已有的词库，仅仅根据词的共同特征，将一段大规模语料中可能成词的文本片段全部提取出来，不管它是新词还是旧词。然后，再把所有抽出来的词和已有词库进行比较，不就能找出新词了吗？有了抽词算法后，我们还能以词为单位做更多有趣的数据挖掘工作。这里，我所选用的语料是人人网2011年12月前半个月部分用户的状态。

成词标准之一：内部凝固程度

要想从一段文本中抽出词来，我们的第一个问题就是，怎样的文本片段才算一个词？大家想到的第一个标准或许是，看这个文本片段出现的次数是否足够多。我们可以把所有出现频数超过某个阈值的片段提取出来，作为该语料中的词汇输出。不过，光是出现频数高还不够，一个经常出现的文本片段有可能不是一个词，而是多个词构成的词组。在人人网用户状态中，“的电影”出现了389次，“电影院”只出现了175次，然而我们却更倾向于把“电影院”当作一个词，因为直觉上看，“电影”和“院”凝固得更紧一些。

为了证明“电影院”一词的内部凝固程度确实很高，我们可以计算一下，如果“电影”和“院”真的是各自独立地在文本中随机出现，它俩正好拼到一起的概率会有多小。在整个2400万字的数据中，“电影”一共出现了2774次，出现的概率约为0.000113。“院”字则出现了4797次，出现的概率约为0.0001969。如果两者之间真的毫无关系，它们恰好拼在了一起的概率就应该是0.000113×0.0001969，约为2.223乘以10的–8次

方。但事实上，“电影院”在语料中一共出现了175次，出现概率约为7.183乘以10的–6次方，是预测值的300多倍。类似地，统计可得“的”字的出现概率约为0.0166，因而“的”和“电影”随机组合到了一起的理论概率值为0.0166×0.000113，约为1.875乘以10的–6次方，这与“的电影”出现的真实概率很接近——真实概率约为1.6乘以10的–5次方，是预测值的8.5倍。计算结果表明，“电影院”可能是一个更有意义的搭配，而“的电影”则更像是“的”和“电影”这两个成分偶然拼到一起的。

当然，作为一个无知识库的抽词程序，我们并不知道“电影院”是“电影”加“院”得来的，也并不知道“的电影”是“的”加上“电影”得来的。错误的切分方法会过高地估计该片段的凝合程度。如果我们把“电影院”看作是“电”加“影院”所得，由此得到的凝合程度会更高一些。因此，为了算出一个文本片段的凝合程度，我们需要枚举它的凝合方式——这个文本片段是由哪两部分组合而来的。令p(x)为文本片段x在整个语料中出现的概率，那么我们定义“电影院”的凝合程度就是p(电影院)与p(电)·p(影院)的比值和p(电影院)与p(电影)·p(院)的比值中的较小值，“的电影”的凝合程度则是p(的电影)分别除以p(的)·p(电影)和p(的电)·p(影)所得的商的较小值。

可以想到，凝合程度最高的文本片段就是诸如“蝙蝠”、“蜘蛛”、“彷徨”、“忐忑”之类的词了，这些词里的每一个字几乎总是会和另一个字同时出现，从不在其他场合中使用。

成词标准之二：自由运用程度

光看文本片段内部的凝合程度还不够，我们还需要从整体来看它在外部的表现。考虑“被子”和“辈子”这两个片段。我们可以说“买被子”、“盖被子”、“进被子”、“好被子”、“这被子”等，在“被子”前面加各种字；但“辈子”的用法却非常固定，除了“一辈子”、“这辈子”、“上辈子”、“下辈子”，基本上“辈子”前面不能加别的字了。“辈子”这个文本片段左边可以出现的字太有限，以至于直觉上我们可能会认为，“辈子”并不单独成词，真正成词的其实是“一辈子”、“这辈子”之类的整体。可见，文本片段的自由运用程度也是判断它是否成词的重要标准。如果一个文本片段能够算作一个词的话，它应该能够灵活地出现在各种不同的环境中，具有非常丰富的左邻字集合和右邻字集合。

“信息熵”是一个非常神奇的概念，它能够反映一个事件的结果平均会给你带来多大的信息量。如果某个结果的发生概率为p，当你知道它确实发生了，你得到的信息量就被定义为-log(p)。p越小，你得到的信息量就越大。如果一颗骰子的六个面分别是1、1、1、2、2、3，那么你知道了投掷的结果是1时可能并不会那么吃惊，它给你带来的信息量是-log(1/2)，约为0.693。知道投掷结果是2，给你带来的信息量则是-log(1/3)≈1.0986。知道投掷结果是3，给你带来的信息量则有-log(1/6)≈1.79。但你只有1/2的机会得到0.693的信息量，只有1/3的机会得到1.0986的信息量，只有1/6的机会得到1.79的信息量，因而平均情况下你会得到0.693/2+1.0986/3+1.79/6≈1.0114的信息量。这个1.0114就是那颗骰子的信息熵。现在，假如某颗骰子有100个面，其中99个面都是1，只有一个面上写的2。知道骰子的抛掷结果是2会给你带来一个巨大无比的信息量，它等于-log(1/100)，约为4.605；但你只有1%的概率获取到这么大的信息量，其他情况下你只能得到-log(99/100)≈0.01005的信息量。平均情况下，你只能获得0.056的信息量，这就是这颗骰子的信息熵。再考虑一个最极端的情况：如果一颗骰子的六个面都是1，投掷它不会给你带来任何信息，它的信息熵为-log(1)=0。什么时候信息熵会更大呢？换句话说，发生了怎样的事件之后，你最想问一下它的结果如何？直觉上看，当然就是那些结果最不确定的事件。没错，信息熵直观地

挖掘新词的传统方法似乎陷入了一个怪圈：分词的准确性本身就依赖于词库的完整性，如果词库中根本没有新词，我们又怎么能信任分词结果呢？

反映了一个事件的结果有多么的随机。

我们用信息熵来衡量一个文本片段的左邻字集合和右邻字集合有多随机。考虑这么一句话“吃葡萄不吐葡萄皮不吃葡萄倒吐葡萄皮”，“葡萄”一词出现了四次，其中左邻字分别为{吃, 吐, 吃, 吐}，右邻字分别为{不, 皮, 倒, 皮}。根据公式，“葡萄”一词的左邻字的信息熵为-(1/2)·log(1/2)-(1/2)·log(1/2)≈0.693，它的右邻字的信息熵则为-(1/2)·log(1/2)-(1/4)·log(1/4)-(1/4)·log(1/4)≈1.04。可见，在这个句子中，“葡萄”一词的右邻字更加丰富一些。

凝固程度和自由程度，两种判断标准缺一不可。若只看前者，程序会找出实际上是“半个词”的片段；若只看后者，程序则会找出诸如“了一”、“的电影”一类的“垃圾词组”。

在人人网用户状态中，“被子”一词一共出现了956次，“辈子”一词一共出现了2330次，两者的右邻字集合的信息熵分别为3.87404和4.11644，数值上非常接近。但“被子”的左邻字用例非常丰富：用得最多的是“晒被子”，它一共出现了162次；其次是“的被子”，出现了85次；接下来分别是“条被子”、“在被子”、“床被子”，分别出现了69次、64次和52次；当然，还有“叠被子”、“盖被子”、“加被子”、“新被子”、“掀被子”、“收被子”、“薄被子”、“踢被子”、“抢被子”等100多种不同的用法构成的长尾。所有左邻字的信息熵为3.67453。但“辈子”的左邻字就很可怜了，2330个“辈子”中有1276个是“一辈子”，有596个“这辈子”，有235个“下辈子”，有149个“上辈子”，有32个“半辈子”，有10个“八辈子”，有7个“几辈子”，有6个“哪辈子”，以及“n辈子”、“两辈子”等13种更罕见的用法。所有左邻字的信息熵仅为1.25963。因而，“辈子”能否成词，明显就有争议了。“下子”则是更典型的例子，310个“下子”的用例中有294个出自“一下子”，5个出自“两下子”，5个出自“这下子”，其余的都是只出现过一次的罕见用法。事实上，“下子”的左邻字信息熵仅为0.294421，我们不应该把它看作一个能灵活运用的词。当然，一些文本片段的左邻字没啥问题，右邻字用例却非常贫乏，例如“交响”、“后遗”、“鹅卵”等，把它们看作单独的词似乎也不太合适。我们不妨就把一个文本片段的自由运用程度定义为它的左邻字信息熵和右邻字信息熵中的较小值。

在实际运用中你会发现，文本片段的凝固程度和自由程度，两种判断标准缺一不可。只看凝固程度的话，程序会找出“巧克”、“俄罗”、“颜六色”、“柴可夫”等实际上是“半个词”的片段；只看自由程度的话，程序则会把“吃了一顿”、“看了一遍”、“睡了一晚”、“去了一趟”中的“了一”提取出来，因为它的左右邻字都太丰富了。

抽词算法的实现

我们把文本中出现过的所有长度不超过d的子串都当作潜在的词（即候选词，其中d为自己设定的候选词长度上限，我设定的值为5），再为出现频数、凝固程度和自由程度各设定一个阈值，然后只需要提取出所有满足阈值要求的候选词即可。为了提高效率，我们可以把语料全文视作一整个字符串，并对该字符串的所有后缀按字

十
十十四是十四四十是四十
十是十十四是十四四十是四十
十是四十
十四是十四四十是四十
十四四十是四十
是十十四是十四四十是四十
是十四四十是四十
是四十
是四十是十十四是十四四十是四十
四十
四十是十十四是十四四十是四十
四十是四十
四是十四四十是四十
四是四十是十十四是十四四十是四十
四四十是四十

图1 对“四是四十是十十四是十四四十是四十”的所有后缀进行排序后的结果

典序排序。

这样的话，相同的候选词便都集中在了一起，从头到尾扫描一遍便能算出各个候选词的频数和右邻字信息熵。将整个语料逆序后重新排列现在所有的后缀，再扫描一遍后便能统计出每个候选词在原文中所有左邻字的信息熵。另外，有了频数信息后，凝固程度也都很好计算了。这样，我们便得到了一个无须任何知识库的抽词算法，输入一段充分长的文本，这个算法能以大致O(n logn)的效率提取出可能的词来。

实际上我们只需要在内存中储存这些后缀的前d+1个字，或者更好地，只储存它们在语料中的起始位置。

对不同的语料进行抽词，并且按这些词的频数从高到低排序。你会发现，不同文本的用词特征是非常明显的。

《西游记》上册：行者、师父、三藏、八戒、大圣、菩萨、悟空、怎么、和尚、唐僧……

《资本论》全文：商品、形式、货币、我们、过程、自己、机器、社会、部分、表现……

《圣经》全文：以色列、没有、自己、一切、面前、大卫、知道、什么、犹大、祭司……

《时间简史》全文：黑洞、必须、非常、任何、膨胀、科学、预言、太阳、观察、定律……

当然，我也没有忘记对人人网用户状态进行分析——人人网用户状态中最常出现的10个词是：哈哈、什么、今天、怎么、现在、可以、知道、喜欢、终于、这样。事实上，程序从人人网的状态数据中一共抽出了大约1200个词，里面大多数词也确实都是标准的现代汉语词汇。不过别忘了，我们的目标是新词抽取。将所有抽出来的词与已有词库作对比，于是得到了人人网特有的词汇（同样按频数从高到低排序）：

尼玛、伤不起、给力、有木有、挂科、坑爹、神马、淡定、老爸、卧槽、牛逼、肿么、苦逼、无语、微博、六级、高数、选课、悲催、基友、蛋疼、很久、人人网、情何以堪、童鞋、哇咔咔、脑残、吐槽、猥琐、奶茶、我勒个去、刷屏、妹纸、胃疼、飘过、考研、弱爆了、太准了、搞基、忽悠、美慕嫉妒恨、手贱、柯南、狗血、秒杀、装逼、真特么、碎觉、奥特曼、内牛满面、斗地主、腾讯、灰常、偶遇、拉拉、九把刀、高富帅、阿内尔卡、魔兽世界、线代、三国杀、林俊杰、速速、臭美、花痴……

我还想到了更有意思的玩法。为什么不拿每天状态里的词去和前一天作对比，从而提取出这一天里特有的词呢？这样一来，我们就能从用户状态中提取出每日热点了！事实上，有了抽词算法之后，我们就能够以词语为单位，在各个维度上挖掘人们在社交网络中的用词动向：今年网络上最流行的新词是什么？男生和女生都喜欢说些什么？不同地方、不同年龄段的人都爱用什么样的词？这些问题曾经需要靠大规模语言统计调查才能得到一个满意的回答，而现在，借助互联网海量的用户生成内容，我们可以轻而易举地得到答案，为社会语言学的研究提供真实可靠材料。受篇幅限制，我们不妨把这些激动人心的话题留到下一篇文章中去。



顾森

网名Matrix67，北京大学中文系应用语言学专业学生，数学爱好者。2005年开办数学博客http://www.matrix67.com，至今已积累上千篇文章，已有上万人订阅。

内容推荐和优化的在线模型

文 / 耿秀波，何力，李伟，赵贤宇

如何设计有效的推荐算法以便实时满足用户对内容的需求，已成为一个亟待解决的问题，并受到研究领域和工业界的广泛关注。本文讨论了为Yahoo！首页进行内容动态优化的C.O.R.E系统（Content Optimization and Relevance Engine）的原理和实现方法。

概述

传统的搜索引擎采用自动排序算法返回符合用户查询需求的结果，并综合考虑了搜索结果的相关性和重要性；而在线广告投放同样采用了类似的技术。相对而言，大部分的门户网站主要采用人工编辑的方式完成内容的展现和组织，主要原因是很难同时从相关性、多样性、新颖性和用户兴趣几方面对内容作出最优选择，另外内容自身的质量也千差万别，目前还没有对内容质量进行度量的统一标准。

人工编辑虽然可以保证高质量的内容，但随着内容和展示页面规模的增加，纯人工的方式往往不可行。而数据驱动的机器学习可以很好地解决规模问题。我们提出了采用人工编辑和机器学习相结合的方法，在满足人工设定约束的前提下，通过优化算法将最佳内容进行排序并推荐给用户。目前该推荐系统（C.O.R.E）已在Yahoo！主页上部署（图1），并能够根据用户属性完成个性化推荐（图2）。

对内容排序的常用方法是采用基于特征的模型，但该模型并不适用于内容经常变化的情况。例如对于新闻推荐来说，其特点是规模小、更新快、文章生命周期短，这和传统的在线搜索和广告投放问题有本质区别，后者的特点是和查询词匹配的相关内容集合在一段时间内相对稳定，无需实时跟踪和更新内容。因此门户内容的实时更新问题对传统的基于特征的预测方法提出了新的挑战，也是本文重点解决的问题。



图1　Yahoo！主页的Today模块



图2　访问visualize.yahoo.com实时获取Today模块上的各种统计数据

问题描述

我们以Yahoo！主页中最受关注的Today模块来描述C.O.R.E所解决的内容优化问题。该模块从多篇

候选文章中选出若干篇进行展示。候选文章集合由专业人员选出，并保持持续更新。C.O.R.E主要解决的问题是如何从候选文章中选出最好的几篇展示给用户，以获得最高的点击率（CTR，即在一定时间内的点击次数与展示次数的比值）。

系统设计上的挑战

从上文提到的应用场景看来，系统设计会遇到非常多的挑战，我们这里权且列举其中最重要的两点。

■ **高度动态的系统**。新闻文章生命周期较短（6~8小时），文章池中候选成员是随时变化的，阅读的用户群体动态增减，每篇文章在每天不同时段、不同位置展示时也会有差异巨大的点击率。我们发现只有对用户点击、阅读的行为建立快速反应的动态模型才能够提高内容展示效果。

■ **大规模的数据处理**。首页大约在每秒有成千上万次的页面访问量，一天访问的用户也有上亿。数据的收集、模型的训练、文章的排序都必须在极低的延迟下完成。例如决定为访问首页的特定用户选择合适的内容进行展示的时间只有几毫秒。

为此，我们花费了大量的精力来建立一个可规模化的系统架构以支持近乎实时的数据收集。用户的点击、浏览行为（称为“事件”）从前端服务器的日志中被挖掘出来并不断地传送到数据收集的集群，该集群可以对用户浏览、点击事件之间存在的时间差进行处理。之后事件流被传送到建模集群。伺服前端的服务器会定期地抓取建模集群更新后的模型，并使用这些模型调整伺服的结果。整个周期，包括数据收集、模型更新和模型发挥效用，仅花费几分钟。

内容优化里面的机器学习问题

人工编辑虽然可以保证高质量的内容，但随着内容和展示页面规模的增加，纯人工的方式往往不可行。因此，这里我们将讨论如何使用机器学习算法自动进行内容优化。一种典型的思路是，使用记录在历史日志中的用户和内容之间的交互数据进行模型训练，然后使用这个模型对未来用户在内容上的点击行为进行预测。在这里，可以使用的用户特征包含年龄、性别、地理位置信息以及根据用户过往阅读过的文章推断出的用户阅读兴趣。另一方面，可以使用文章的URL、类别（如财经类、体育类、娱乐类等）和文章中的关键词作为表示文章的特征。但在实际应用中这种方法的效果并不是很理想。究其原因，用户与推荐内容之间的交互是一个非常动态化的过程，这导致很难用一个在过往历史数据（例如一周前或一月前的数据）上训练出来的模型对用户的行为进行预测。我们发现，即使在同一类别的文章，例如同样是财经文章，不同文章的用户点击率也有很大的区别，而且一篇文章的点击率也是随着时间

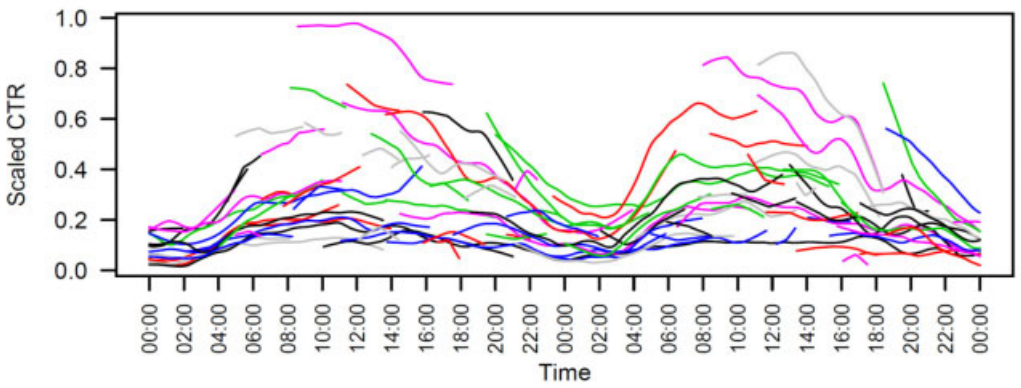


图3　文章的点击率随时间动态变化，每条曲线对应一篇文章的点击率随时间的变化

变化的。下文中，我们将展开具体讨论一些在内容优化问题上机器学习算法所面临的问题和挑战。在本文后面，我们将具体讨论如何使用一些快速收敛的在线模型对用户和内容的动态交互行为进行更好的预测。

■ **非平稳的点击数据**：从图3中，我们可以看出在Yahoo！首页上，不同文章的点击率随时间的变化，图中每条曲线对应一篇文章的点击率随时间的变化。一篇文章的点击率依赖于很多因素，例如它所展示的位置、出现的时间段等（例如工作时间、休息时间、白天或晚上）。同时，一篇文章在它的生命周期内从上线，到引起人们的注意导致点击率提高，进而到被人们阅读后点击率下降，这也是一个动态的随着时间演化的过程。对于不同的文章，点击率会表现出不同的随时间演化的方式。面对这种非平稳的数据，传统的那些适用于平稳过程的算法就难以适用了。为此，我们需要针对每篇文章构建一个动态的在线模型来跟踪、预测这篇文章与用户交互的方式。这比传统的离线模型取得了更好的性能。

■ **数据中的偏歧性**：系统所采用的伺服策略决定了展现哪些文章链接给用户。通常，这种伺服策略（例如编辑的选择、某种内容推荐算法等）会根据一定的准则优先选择某些文章展现，而较少展示另外一些文章。因此，如果直接使用这些用户和内容的交互数据进行模型的学习，将会不可避免地受到目前系统所采用的伺服策略的影响。例如，对于那些系统频繁展示的文章，模型可以较好地估计它们的点击率；而对于那些很少或根本没有得到展示机会的文章，模型则很难或无法做出预测。这种由于伺服策略导致的数据中的偏歧性（bias），对如何有效构建一个机器学习模型提出了新的挑战。我们必须使用某种随机采样策略来构造模型训练数据，以此来减少数据的偏歧。

■ **自动内容优化与网站业务逻辑**：对于类似于Yahoo！的内容网站，在选择展现的内容时通常需要满足诸多的业务逻辑的限制（例如专业编辑人员的意见、某种类别的文章链接不能展现超过某个预设的限制、去重等），因此在最终的展现结果上，我们必须综合根据机器学习算法得到的文章和这些业务逻辑的限制，找到一个最优的排序结果。

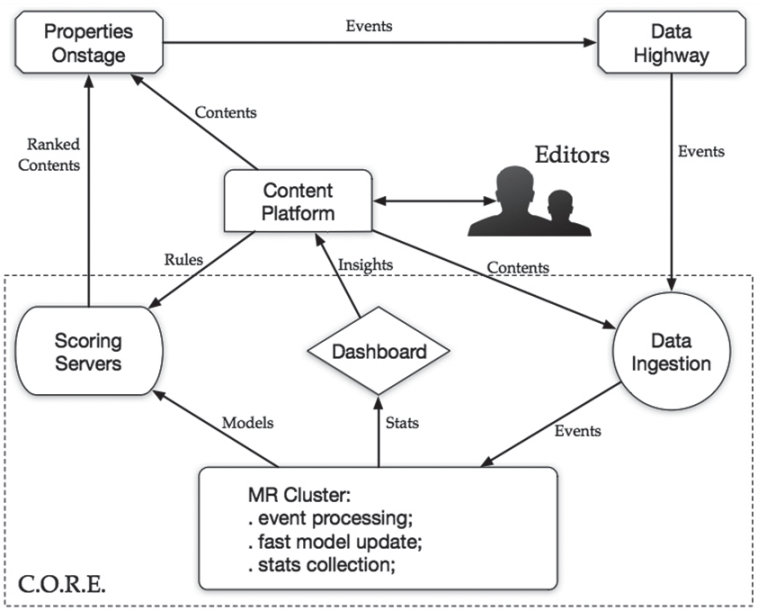


图4 C.O.R.E.系统架构

系统的实现

我们的实现包括以下模块。

■ **批量学习**。通常页面的浏览和后续的点击之间相差2~10分钟，工程上为了处理高流量数据会带来一定的限制，我们的模型大约5分钟更新一次，我们伺服的算法必须能够在这个时间内完成。

■ **业务逻辑，编辑重写排序结果**。尽管算法导向的排序是最终的目标，但现在仍然有一些需要编辑干预的情形，例如突发性具有重大影响力的新闻必须迅速地占据头条位置，一个60分钟重新访问的用户不应该看到曾经看到过的新闻等，这些约束条件可以由编辑人为地加入到此模块中，并且伺服前端的服务器必须满足这些要求。

■ **点击率的在线模型**。通常我们会设计并实现若干个不同的模型来跟进当前每篇文章的点击率，

这些模型可以预测出每个用户种类对应的点击率，绝大部分用户都会享受到根据这些模型预测出来点击率排序过后得到的新闻信息。

■ **E&E策略**（Explore and Exploit）。我们需要少数一部分流量用来发现文章池里面高质量的部分，剩余的流量利用这部分提高用户的点击率。在前面一部分流量中，文章池里面的每一篇都是等可能的被展示给用户，利用它，各种模型就能获得对点击率无偏的估计。系统架构如图4所示。

在线模型

C.O.R.E系统中采用了三种在线模型：EMP（Estimated Most Popular）模型、SMP（Segmented Most Popular）模型，以及OLR（Online Logistic Regression）模型。

EMP模型是一种基本的计数模型，其基本思想是，对每篇文章利用历史数据中展示给用户的次数以及用户点击的次数来估计该文章的点击率进行估计。假设每五分钟更新一次模型，那么我们估计所得的点击率为：

$$CTR_{EMP} = \frac{C_t + \gamma C_{t-1} + \gamma^2 C_{t-2} + \ldots}{V_t + \gamma V_{t-1} + \gamma^2 V_{t-2} + \ldots}$$

其中C_t表示t时刻的点击次数，V_t表示t时刻的展示次数。EMP模型的优点是计算简单，易于实现；但它没有考虑到不同用户对同一文章的点击率不同，例如男性用户比女性用户对体育新闻的点击率更高。

SMP模型对EMP模型进行了改进，它将用户根据其属性进行分类，对每类用户进行文章点击率的估计。例如，男性用户的点击率为：

$$CTR_{SMP_Male} = \frac{C_t^{Male} + \gamma C_{t-1}^{Male} + \gamma^2 C_{t-2}^{Male} + \ldots}{V_t^{Male} + \gamma V_{t-1}^{Male} + \gamma^2 V_{t-2}^{Male} + \ldots}$$

当描述用户的属性增多时，每类用户获得的数据大幅减少，使得SMP的估计失准。此时OLR成为一种更自然有效的模型。我们用x来表示描述用户的各种属性，例如x=（年龄、性别、地理位置等），OLR模型认为用户点击文档的概率为：

$$P(click) = \frac{1}{1 + \exp(-\sum_f w_f x_f)}$$

其中w_f是对第f个属性的权重。OLR模型有效地考虑了用户属性对点击率的影响，并且可以充分利用历史数据进行学习；相应的，对OLR模型进行更新的复杂度也更高。

结论

本文讨论了雅虎为用户提供个性化新闻服务的内容优化平台的一种实现形式，感兴趣的用户可以访问visualize.yahoo.com实时获取各种统计数据，体验C.O.R.E如何为各种人群提供相关、有趣的内容推荐。

在目前的基础上，一个很自然的想法就是将其应用到其他问题上，例如雅虎财经、体育等不同的频道上的个性化内容推荐。另外，点击率优化并不是一个唯一万能的准则，更多的时候我们还需要考虑均衡用户的体验与公司的盈利。

另一方面，目前C.O.R.E采用的这种模式也有一定的局限性。当候选文章数目较少、流量巨大时，我们能够获得每篇候选文章点击率较为充分的估计，这使得我们现有E&E策略得以实施。当候选文章较多、流量相对偏少时，观测的稀疏可能会导致现有模型不准确的估计。换用其他的E&E方式在某种程度上能够减缓这个问题，但可能仍然不是一个根本的解决方案。我们需要探索新的模型和学习方法。

（本文四位作者均来自雅虎北京全球研发中心）

“人工+智能”：机器翻译应用的未来

文 / 魏勇鹏

一段时间以来，网络上关于机器翻译的争论引起了越来越多的关注。或许是微博等社会化媒体的流行使得令人啼笑皆非的机器翻译错误被更充分及时地曝光了，又或许是百度翻译的上线激发了部分国人PK Google的某种趣味，总之，一时间各种调戏、对比，各种“神”翻译、“雷”翻译层出不穷。

其实机器翻译是一个很古老的话题了，本文仅以机器翻译研发的从业人员角度，谈谈对机器翻译应用前景的看法。

当我们谈论机器翻译时，我们谈些什么？

首先我们尝试从机器翻译圈子多年来的话题——“标签”的变迁中来寻找一些关于发展趋势的蛛丝马迹。本文就不追溯到机器翻译问世的20世纪中叶了，仅以我进入机器翻译领域的近十年为限，作为管窥之见抛砖引玉吧。

2002—2006年

【标签云内容：“统计机器翻译”、“统计还是规则？”、“模板”、“BLEU、BLEU还是BLEU！”、“层次短语”、“串到树-树到串”】

统计方法方兴未艾，学术界一片乐观景象，大有“只要给我足够的语料，机器翻译的突破指日可待”的架势。不过在工业界，占据市场主流的还是传统的规则方法或模板/实例方法。两类方法不可避免地引起了一些争论，于是就有了“规则是傻子，统计是疯子”的各种比喻。不管怎样，伴随着统计方法的兴起，机器翻译研究的门槛确实降低了不少，各种机器翻译评测的参与热情空前高涨。在评测中普遍应用的指标就是BLEU值，也就是看机器翻译结果和人工翻译结果（通常有多组）有多大比例的N元词组是一致的。这种评测方法虽然看起来简单粗浅，但在系统对比方面却十分有效。统计模型的研究逐步复杂深入，从单词、短语到层次短语，到引入句法树的“串到树/树到串/树到树”模型。

2007—2011年

【标签云内容：“统计机器翻译”、“多系统融合”、“统计+规则”、“BLEU之外的评测指标”、“限定领域”、“语料清洗”、“增量训练”、“译后编辑”】

国内的搜索引擎厂商如有道、百度等先后推出了在线翻译引擎，统计方法渐渐占据主流。但单靠统计模型的改进或语料规模的扩张已难以取得突破性进展。统计方法必须融合语言学规则乃至更广泛的专业知识库、本体库，这已成为学界共识。但融合什么、怎样融合，怎样低成本高效率地建设所需的资源，都还处在探讨阶段。评测指标方面，不再唯BLEU马首是瞻（对参与评测的热情也开始降温），一些更具实际操作意义的指标，例如译后编辑中的编辑距离（即将机译结果调整到人译的标准所需的工作量）开始得到重视。让机器翻译在合适的领域先应用起来的思想推动工业界出现了更有价值的应用，例如2010年底Google与欧洲专利局合作推出的欧洲语言之间的专利翻译引擎；也开始出现为企业提供翻译引擎定制、语料清洗加工、模型训练调优的专业服务提供商，例如总部位于泰国的Asia Online（我



们也属于此类服务提供商）。

2012年—

【标签云内容：“统计机器翻译”、“语义”、“译后编辑”、“客户定制”、“用户反馈”、“领域适应性”、“众包模式”、“人本机器翻译”、“云翻译”……】

目前来看，统计仍会是主流，但在3~5年的短期内在算法模型方面取得根本性突破的可能性不大。不过加速机器翻译技术转化为实际的产业应用已然是大势所趋。各个互联网巨头都加大了在这方面的投入。更重要的是，开源系统的成熟进一步降低了研究和应用的门槛。例如由欧盟资助的MosesCore开源机器翻译项目正在全球不遗余力地进行推广。此外，云计算的火热也催生了“云翻译”。可以预见在各行各业即将涌现出成千上万种根据领域乃至客户定制的翻译引擎。具体应用模式上，译后编辑（即在机器翻译的结果上进行人工编辑）在欧洲语言之间的翻译中取得了不错的效果，不过在差别较大的语种之间仍不理想。用户参与、用户体验、用户反馈在产品中越来越受重视（例如谷歌在线翻译加入了大量用户修正或反馈的互动），基于大规模群体智慧的“人本机器翻译”（此概念为国内机器翻译研究领域的泰斗董振东先生提出的）也许正是突破口之所在。

从上述标签的演变可以看出，我们越来越清楚地意识到，实际“用起来”才是最重要的事情。然而，用户并不关心你的原理是基于“统计”还是“规则”，不关心你用的是什么模型什么算法，也不关心BLEU之类的评测指标，那么他们关心的是什么呢?

机器翻译真的会取代人工翻译吗?

这可能是一部分从事语言服务的用户会关心的问题。2011年，曾有一位“未来学家”Ray Kurzweil预言：到2029年，机器翻译将达到人类翻译的水平。尽管这位预言家也认为人类翻译并不会因此而失业，正如电子音乐合成器没有让音乐人失业一样，但仍有不少从业人员对此抱有担心甚至抵制的心态。

其实正如预言家所说，翻译是“人们可以想象的最高水平的工作”。成为一个专业的翻译人员，除了对源语言和目标语言双重精通的要求，还需要对翻译工作面向的行业有尽可能准确深入的认识，而且还要对目标语言的受众群体的心理特质有足够的理解。一般而言，要成为一个靠谱的

专业译者，没有七八年的专注实践经验是不行的。因此，莫说是机器，就是人类本身，不经过足够的努力，要想“取代”这样的专业译者都是不可能的。可能还有人会说，你说的只是少数的“高端”翻译不会被取代，那么大多数中低端的从业者呢？其实只要不是对技术持着畏惧和排斥的态度，而是反过来充分利用技术和工具带来的便利，就可以大大缩短达到“高水平”的时间，完全可以跑赢那些看起来要“取代”你的机器。

从另一个角度来看，自动翻译技术并非仅会蚕食现有市场，同时也会创造更大的市场。根据语言服务行业比较权威的调研机构Common Sense Advisory的一份报告，目前企业中只有1%的语言内容得到了翻译。这些内容，要么是最高端的，例如企业最重要的产品、市场、财务资料，一般交给专业的语言服务提供商来完成；要么就是最低端的，让用户自己利用在线翻译引擎简单了解即可。而其他99%的内容还是未翻译的，主要包括企业经营中实时产生的数据、即时客户服务的内容、用户生成的内容以及历史积累的各种知识库等。这些内容规模很大，完全由人工翻译的成本太高，可是质量要求却又不低，不能用自动翻译简单处理了事，因而只能暂时先放着不做处理。随着机器翻译技术的发展和应用，可能会出现一些新的工作模式，能够将人工翻译的高质量与机器翻译的高速度结合起来，共同开垦那99%的广阔疆域。

那么这些可能出现的新工作模式是什么样的呢？

人工智能 = 人工 + 智能？

有人说，机器翻译可能是人工智能的第一座堡垒，但也可能是最后一座堡垒。也许我们在各种科幻故事或电影中所理解的人工智能，就是想要“取代”人工的机器智能。然而现实中取得成功的人工智能，基本都不是这样的对立关系，而是“人工”+“智能”的共生关系。

具体到机器翻译上。其实机器翻译发展的各个阶段，都可以认为是对人工的利用和支持过程。只是每个阶段的人工内容不太一样。

规则阶段

人工内容是语言专家撰写的语言规则。显然这是只被少数精英人士“代表”的人工数据。然而语言是复杂、不规范且随时变化的，而且当时还没有足够的语料库，很多规则是依靠专家“内省”得到，这样的人工内容其实很难充分描绘真实的语言世界。

语料库前期阶段

人工内容是语言工程师在语料库上所做的标注或者根据语料库编制的各种模板。因为语料库的引入，这些内容比较接近真实的语言世界了。但人工标注容易出现因标准不统一造成的数据不兼容，而且标注工作费时费力（对标注者要求也很高），因而数据规模和质量都比较难保证。

语料库后期阶段

人工内容是由语料库工程师建立的大规模平行语料库。随着互联网及搜索引擎技术的发展，大规模真实双语文本的采集和利用成为可能。机器翻译系统使用统计方法在这些直接的语言材料中“学习”翻译的模型。这些双语语料的创造者是长期以来大量的语言工作者。因此，这是目前为止对真实语言资源的最充分利用。

人机交互阶段

然而上面所述的双语语料库仍然只是静态的结果文本，并没有语言工作者在创作或翻译过程中的动态数据，所以上面那样“学习”出来的模型跟人类学习和理解的语言体系并不是一回事。译者在翻译过程中的动态行为，例如先翻译哪个词后翻译哪个词、译词的选择和修改、所查询的术语及采取的操作、原文理解的时间和视线焦点分布、与项目组成员（项目经理、校对人员、客户、其他译者等）的沟通内容、对翻译记忆库或机器翻译结果的利用情况、所参考的资料等，如能得到大量译者的行为数据并充分挖掘有用的信息，那将使机器翻译的过程及结果产生革命性的变化。这才应该算是机器翻译领域真正的“大数据”时代——我们所理解的大数据，不单是语料规模大，更重要的是数据的实时性和全面关联

性。我们正在迎接这个时代的到来。大幕将如何拉开呢?

计算机辅助翻译：人机合一的切入点?

计算机辅助翻译（CAT）顾名思义就是由计算机程序提供辅助以提高人工翻译效率及质量的过程。在这个过程中，人是翻译活动的主体，机器是辅助手段，因而天然就是人机互动的最好契合点。在机器翻译质量还远不能令人满意且个人计算机的运算能力有限的条件下，计算机辅助翻译主要解决的是降低重复劳动率的问题，即通过“翻译记忆”的方式，将以前翻译过的句子存储起来，在翻译过程中检索到相似的句子及译文就可以进行复用。这种方式的互动较少。后来机器翻译在某些语言对之间（比如欧洲语言间）取得了较大的进展，于是前面提到的“译后编辑”方式开始占领舞台，这种方式的互动就较为深入和频繁，但机器和人工存在一个“先”和“后”的次序，实时性不足。

国外CAT产品的应用已比较广泛，主要有SDL Trados、Déjà vu、Wordfast等。基本上大中型的语言服务提供商（LSP）都会部署多套CAT系统，这些LSP的客户方通常也对CAT的使用有明确的要求并会提供翻译记忆库（TM）、术语表等相应资源。

而在国内，CAT应用的比例还非常低，大多数翻译工作者还是采用纯手工翻译的方式。一方面，国内的LSP及其客户企业的信息化技术水平都还比较低；另一方面，因为语种及工作习惯的关系，现有的CAT产品（不论国外产品还是民族产品）都不能满足用户的需要。正如一个未经训练的人上来就开手动挡汽车，自然是一件非常困难的事情。不仅如此，操作不便的手动档汽车类型繁多（产品交互体验不好）；市场上还充斥着改装的走私进口汽车（盗版严重）；驾校师资不够；教学手段匮乏；教学内容浅薄；教学学时不足（翻译专业教育中对技术重视不足）；学车人员排斥练车；宁愿步行（译者主动学习积极性低）；鲜有的学成者出来驾驶营运车辆，却因驾驶技术不熟练，乘客投诉不断（行业接受度不高）等。这就是当前国内计算机辅助翻译工具的应用现状。

虽然国内的CAT应用现状还不理想，但这也意味着极大的发展潜力：1. 越来越多的国内语言服务提供商已经意识到机器翻译技术将对行业产生显著影响，并积极尝试加以利用；2. 国内语言服务行业正在酝酿着重大的结构调整和资源整合，技术将在此过程中扮演重要角色；3. 云计算、社交网络的兴起在语言服务行业催生了新的商业模式，如云翻译服务平台、众包模式等；4. 将会出现更多专注于特定垂直领域的特色语言服务商，有利于定制化的机器翻译引擎发挥作用；5. 中文及中国市场在语言服务行业中的份额将显著增加；6. 用户受现有CAT 产品的交互模式影响小，迁移代价低，可以更容易接受更为深入和全面的互动模式。

目前我们正在研发一种互动式的机器辅助翻译引擎，在人工翻译过程中根据用户的各种行为及决策意图，给出实时的输入提示和翻译建议。在这个过程中，人做的是人擅长的决策和调整工作，而机器完成的是机器可控的统计和推荐，两者各司其职、各取所长，可以最大限度地提高辅助翻译的效用。通过云服务平台，大量的优质群体智慧将得到聚合和挖掘，使得“人”与“机器”、“人”与“人”之间不仅是“相加”的关系，而将会创造出更多的价值。愿与各位有志于机器翻译研发及应用的同行，以及语言服务行业的朋友们，一起见证翻译技术及语言服务产业的新发展!



魏勇鹏

北京语智云帆团队联合创始人，语言智能技术服务方向创业中。曾在清华大学、同方知网从事多年机器翻译研发。译有《写给大家看的搜索引擎营销书》。

基于HMM的中文整句输入法实现

文 / 卿锋

开发起源

2003年时我还是一名Unix程序员，Mac OS X的Darwin核心吸引了我，于是购买了一台有128MB内存、G3 800处理器的iBook G3。作为一个中文用户，日常使用Mac OS X时遇到的第一个问题就是内置中文拼音输入法很难用，与当时Windows上流行的紫光、拼音加加等输入法完全没法相提并论，不能对用户输入实时反馈，没有模糊拼音，没有双拼，极其匮乏的词库……更糟糕的是还没有任何第三方的输入法可以使用，朋友用过以后打趣说，Mac OS X有最华丽的界面和最难用的输入法，以至于只能看没法用。

当时国内的Mac开发界圈子很小，国产第三方软件几乎没有。我在渤麦堂（当时国内最大的Mac开发网站）里认识了几位Mac开发爱好者，互相交流学习的过程中渐渐对Mac开发产生了兴趣。虽然工作很忙，但每次打开笔记本，输入法的事情就直接堵在我的眼前，提醒我这个平台的输入法还停留在远古时代。

心里觉得堵得慌，就在工作空闲期间研究起苹果提供的一个例子BasicInputMethod，正如名字所言，真的很Basic，要实现中文拼音的输入，还需要很多的改进。而苹果提供的开发文档也糟糕，大多数情况下都是不断摸索尝试着去理解API的含义，网上的资源也极其有限，当时真有孤立无援的感觉。

进步与发展

在做QIM之前，我对输入法的开发一窍不通，出发点也很单纯，只是为了让自己能无障碍地输入中文。但随着有越来越多用户开始使用它，以及自己知识的积累，我发现开发一个输入法绝易事。

一个拼音输入法最基本需要支持的功能有：双拼、词库管理、模糊拼音、热键管理、特殊符号录入、拼音纠错等。除此之外，还要为高级用户准备一些特殊功能，比如辅助码、音调选择、自定义字符、繁体输出等。从现在QIM庞杂的设置选项里，就能看出当初为适应用户需求所做的工作。

词库对于拼音输入法来说非常重要，个人搜集的词库已经无法满足用户的需求，虽然QIM可以通过外挂词库增加词汇量，但不是每个用户都会这么做。通过努力，我争取到了Sogou的词库授权，应用其备受好评的基础词库以后，QIM的输入流畅度得到了显著提高。后来又得到了搜狗细胞词库的支持，这极大增强了QIM的可用性。

输入法领域还有一个比较有“技术含量”的技术，就是整句输入。为了让QIM能够做到整句识别，我特意购买了黑马神拼的整句技术，融合进入QIM后能够让用户体验到不错的整句识别，但由于黑马的数据库有些年头，在应对新的网络语言时会力不从心，而且数据库也过于庞大，难于下载。但通过黑马的代码，我了解了NLP（自然语言处理）领域的一些知识，这也为我后面对QIM发展做了很好的铺垫。

这里要特别感谢输入法界的一位前辈，在了解我打算进一步完善QIM的整句以后，倾囊相授他了解的HMM模型工程实践及算法方面的知识，正是有他的指导我才能完成这么一个看起来对于

个人几乎不可能完成的工作。我就把这次实践的流程做一次介绍。

准备工作

要实现一个HMM模型首先需要海量的汉字文本作为基础语料，收集这么多语料是一个很艰苦的工作，幸运的是搜狗提供了免费的文本资料用户使用（容量2TB左右），大概有400多亿汉字。但这些文本质量不是很高，主要因为内容是从网页抓取的，而网页存在重复引用问题，尤其是某一段时间里的热点新闻会被大量重复。此外，很多论坛、新闻网页里存在一些运营商链接，会导致某些句子或者关键字被大量重复，这会最终导致模型输出意想不到的结果，不过作为学习用途已经够用。此外你还需要拥有一台配有大容量硬盘、大容量内存的强劲的机器来帮助你处理海量文本数据。否则，为了得到结果，可能要经过漫长等待。

预处理

为了方便后继处理，文本首先要经过预处理，一般来说，理要经过以下几个步骤。

剔除不必要的文字和符号

首先统一文本的编码，我把所有文本都统一为GB18030，网页里所有UTF8、BIG5字符，都要转换为相同的编码（代码需要搜索<meta>里的encoding信息，然后再将文本转换好，但不能完全相信encoding内容，有时会转换失败的），这里有个小技巧，如果发现连续的非GB2312文字出现，就可以认为文本是乱码了，因为普通文本里连续出现非GB2312文字的概率是很低的）。

其次，网页里注释的内容要剔除，如果发现一些广告链接也要尽量剔除，只保留主要的文本。

第三，剔除文本里的一些特殊符，一般情况下汉字输入只会用到很少的标点符，基本上符号保留最常用的汉字标点就可以了，可以统一半角和全角标点，例如罗马标点“,”和中文标点“，”可以视为一个标点符号。还有如果发现网页文本里存在某些注释、广告等都要剔除。

合并字母和数字

一个数字或者字母和多个数字和字母在语义上是一样的，因此可以把文本里的数字和字母进行合并，例如“2008年12月”和“销售了3025.32万元”，可以合并为“D年D月”和“销售了D元”，其中D就代表数字。

添加句头和句尾标示

句子的开头和结尾都需要添加特殊的标识（这里句头用HEAD表示，句尾用EOF表示，当然为了今后程序更容易处理，建议用2个字符的字母表示，这里只是为了更容易理解），因为它们在模型转换过程中都是需要参与计算的。预处理的结果我们期望得到一组标准化后的文本，文本里只出现汉字、标点和其他已经转义以后的特殊意义字符，然后以句子的形式保存在文本里，例如类似下面的结果：

HEAD国家财政部最近下发了D年财政资金D亿元，用于补贴…，帮助企业走出困境。EOF

波兰作曲家EN的交响乐深深打动了观众，在连续D小时的演奏中….

标注拼音

这是一个很重要的步骤，汉语里多音字居多，例如“灯”的ding发音，会把“ding'huo”（订货）误判为“灯火”，“车”的ju发音，会把“ju'xi”（据悉）误判为“车系”，由于它们频率很高，会严重影响最终结果。标注结果需要和原始数据做出一一对应，例如：“波兰作曲家EN的交响乐深深打动了观众”标注后可以为“bo'lan'zuo'qu'jia'D'de'jiao'xiang'yue'shen'shen'da'dong'le'guan'zhong'.”，这样可以方便下一步切分。

切分与统计

这个步骤中我们需要统计所有文本的一元组、二元组、三元组的出现次数。

一元组的统计，就是对汉字、标点以及特殊符号的统计。对Sogou的语料库统计结果，单字出现频率头10位为：

汉字	次数
的	3436620471
一	1330370899
网	1285398388
人	1279143810
中	1184976303
大	1018670690
新	981265843
有	960256861
我	956678857
不	953161411
在	886335963

这里就“网”字比较特别，竟然频率这么高，可能是由于数据都是网页里抓下来的缘故。

二元组的统计就是把文本里的内容，每两个字符一组，依次统计下来即可，例如“HEAD今天天气很好！EOF”，按照二元组切分可以得到如下的结果：

HEAD今
今天
天天
天气
气很
很好
好!
! EOF,

三元的统计就相对比较困难了，倒不是算法上有什么特别的地方，而是在于三元组的数量太庞大了，要有效率地处理这些三元组是一个挑战。经过实际比较Zend公司的PHP里的Hash算法效率和内存使用率都是比较好的，建议大家使用。

算法

其实HMM算法在好多地方都有比较详细的说明，我特别推荐Google黑板报里“数学之美系列”中的“统计语言模型”和“隐含马尔可夫模型在语言处理中的应用”。

公式

如果S表示一连串特定顺序排列的词w1，w2，…，wn。换句话说，S可以表示某一个由一连串特定顺序排练的词而组成的一个有意义的句子。现在，机器对语言的识别从某种角度来说，就是想知道S在文本中出现的可能性，也就是数学上所说的S的概率用 P(S) 来表示。利用条件概率的公式，S这个序列出现的概率等于每一个词出现的概率相乘，于是P(S) 可展开为：

```
P(S) = P(w1)P(w2|w1)P(w3| w1 w2)…P(wn|w1 w2…wn-1)
```

公式很晦涩难懂，简单地说，“隐马尔科夫模型”是一种“透过表象猜本质”的模型，在输入法里，所谓“表象”就可以理解为拼音串的输入，而要“猜”的“本质”就是隐藏在背后的中文的输出串了（也就是其“隐”的由来）。由于汉语的多音字实在太多，程序“猜”的时候光靠当前的拼音显然是不行的，我们得借助前面已经“猜”出来的汉字才行，如果借助前面一个汉字的结果(P(w2|w1)，可以称为“二元”模型，如果借助前面两个汉字的结果，可以称为“三元”模型(P(w3|w1w2))，当然如果想提高“猜”的成功率，可以往前借助更多的汉字，构造出“四元”甚至“五元”模型来，但需要的资源就更加庞大了。基本上当前主流输入法模型都开始以三元模型为主。

算法

假设我们输入了一个句子的拼音“hao tian qi”（好天气），这组拼音串的组合出来的汉字概率用P(S)来表示，可以得到：

```
P(S) = P(hao)*P(tian|hao)*P(qi|hao-tian)
```

这里P(hao)就是hao拼音的一元统计概率，P(tian|hao)就是tian拼音的二元统计概率，而P(qi|hao-tian)就是qi的三元统计概率，如果句子更加长，那后面的字就依然使用三元统计概率计算。

一元统计概率P(hao)就是拼音“hao”对应汉字在数据库中出现的概率；

二元统计概率P(tian|hao)可以表示为P(tian|hao) =a*P(hao-tian/hao)+b*P(tian)

（其中a、b是参数，a+b=1.0，如果a=1.0，b=0，那么这个二元模型就是基于词组的输入模式了）

由此拓展到三元统计概率P(qi|hao-tian)可以表示为P(qi|hao-tian) = a*P(hao-tian-qi/hao-tian) + b*P(tian-qi/tian) + c*P(qi)

（其中a、b、c是参数，a+b+c=1.0）

一个拼音下对应的汉字是很多的，上面的例子中hao下的汉字有60多个，要得到整个句子的概率，必须每个汉字都要计算一次，这样下来的路

径的数目就非常庞大了，假设拼音个数为T，每个拼音的汉字是N，那整个计算复杂度是$2TN^t$，还好有一些算法可以帮助我们在合理的时间内找到较好的路径（可以参考Viterbi算法）。

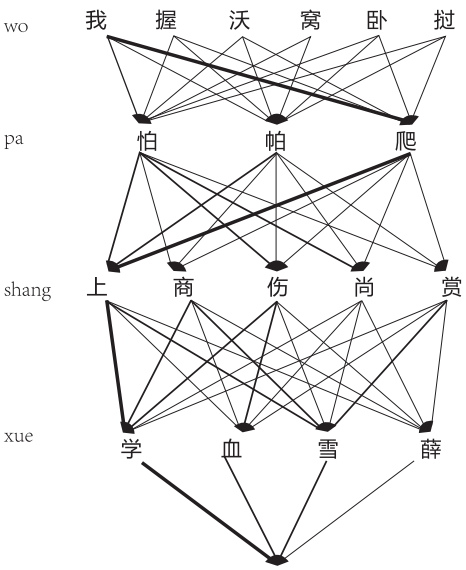


图1 路径中P(S)值最高的就是我们要呈现给用户的结果

调试

简单按照算法得出的结果往往不是最优的，需要对公式里的参数做一些调整。例如找一些语料，设定a、b、c的参数，然后用语料进行转换训练，找到最高的成功转换率。

瘦身

GB级别的数据库显然无法部署给用户，需要对数据库做大幅度删减。如何最小限度地减少因为数据量减小而导致识别率下降是一个需要注意的问题。

最简单的办法，是把数据库中低于某一次数的元组剔除，保留出现频率高的元组，这个方法虽然简单，但会让识别率下降较多。

另外一个办法就是用现有的数据库用一组语料来进行训练，对成功识别的每个句子计算出一个评估值（评估值可以用成功转换句子的概率除以句子的长度，也就是P(S)/N），然后再用该模型对所有的语料重新进行识别，如果低于这个评估值的句子，就不再进入下一步的训练使用（因为可以认为，即使该句子进入语料库也不会提高模型的成功率）。

这个方法可以最大限度保留模型的成功率，但代价显而易见，要重新对所有语料进行转换是一项非常庞大的工作。

瘦身后的数据，成功率有少量降低，但是值得的，毕竟太大的容量没有办法交付到用户手中。

总结和展望

如果有朋友曾经关注过我的Blog，就知道我前段时间在做三元马尔科夫模型的实践，并且已取得了成功，这个是我做输入法的终极目标，今后我会把这个模型移植到QIM上，让用户体验到更好的整句输入。

其他功能上，我想以后通过把QIM的一些功能外挂化来提供给一些有兴趣的开发者以拓展输入法的功能，毕竟如果只是一个人开发，进度上难免滞后。

目前各个大公司的中文输入法在三元模型的应用上都已成熟，基本是在拼外围功能了。我感觉搜狗的词库目前最为完善，而且联网使用时会有云结果，这个模型比三元HMM模型的结果更加完美，可能是动用了更高阶的模型，只是速度还不很流畅。我期待中文输入法以后可以引入语境识别技术，让输入法能够识别用户的意图，加载合适的数据库提高成功率。



卿锋

1999年毕业于中南大学计算机科学系，2003年开发了Mac OS X下的第一款第三方中文拼音输入法QIM，并一直不断持续更新至今，目前已开始专注输入法的智能整句输入。

轻敏捷

开放平台的制胜之道

文 / 王晓明

开放平台下的开发具有不确定性强、可变性强等特性，而轻敏捷开发方法能极大地帮助团队应对这些特性，在激烈的竞争中取胜。本文结合轻敏捷方法的三条原则，讲述敏捷团队所需要完成的工作、核心实践及所依赖的工具。

开放平台下的互联网和游戏开发要求团队更快速地对用户反馈做出反应，快速地验证设计是否合理，开发中更少的浪费，更多的工作需要由团队自组织完成而不是等待项目经理或团队负责人来做决定。由此可以总结出适合这一需求的、更轻量级的软件开发方法——“轻敏捷方法”的三条原则是：快速验证、价值驱动及自组织战队。那么，怎样才算实现了这三条原则呢？

- 团队建立了清晰、简单的目标，并用科学的方法获取需求、验证设计。
- 团队建立了简单易用的迭代流程。
- 项目信息完全可视化。
- 团队通过持续不断的工作回顾来优化和改善项目绩效。

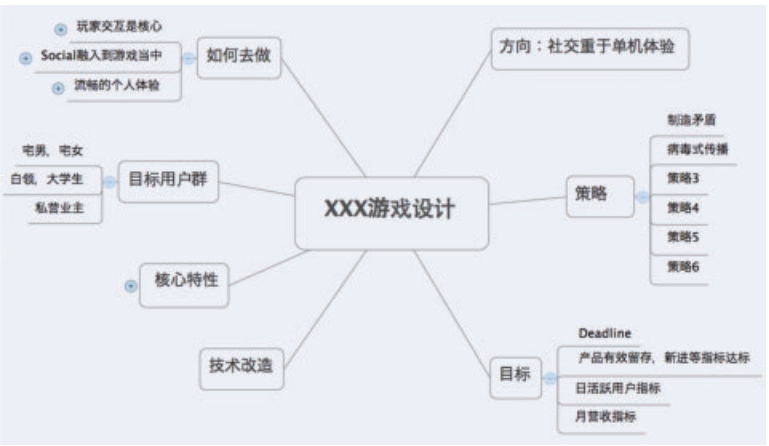


图1　建立清晰简单的目标

本文将结合轻敏捷方法的三条基本原则，讲述敏捷团队所需要完成的工作、核心实践及所依赖的工具。

为团队建立清晰、简单的目标

即使是人数很少的团队，其产品失败的主要原因都可能是团队对产品和项目的目标理解不一致，有的团队甚至连个清晰的目标都没有，团队成员各自为战。尽管每个人都很努力，但目标不同，向不同的方向努力只会造成项目中大量的浪费。每个成熟的团队都会在项目初期或项目的重要转折点使用头脑风暴的方式来共同产出团队目标。但仅有目标还不够，还需要做到以下几个方面。

- 一句话描述产品方向。
- 设计策略。
- 实现方法（可能是具体的特性）。
- 目标用户群定义，用户画像。

图1是从一个真实案例中提取的产品分析模板，以供参考。

用科学的方法获取需求，验证设计

应用开发是从产品需求获取开始的。由于针对开放平台的多数应用规模不大，所以用户期望使

用你的产品在较短的时间内解决问题。作为开发者，首先你必须是应用的第一个用户，并不是说第一个使用产品的人，而是第一个切身感受到这个产品的存在必要性的人。因此当你有灵光闪现时，需要勾勒出产品，但不要急于设计，应该先描述产品的特性。可以使用“A Product in Box”的方法，将产品的内部看做Black Box（黑盒），只关心产品的外在特性，同时回答以下问题。

- 该产品解决了用户的哪些问题？例如，让一个MM无须看上百个网页去找心仪的裙子，只搜索关键字就好，或者让一个无聊的人可以在虚拟世界里得到现实中无法获得的成功，从而更快乐地生活。
- 如果不解决这些问题，用户会感到痛苦吗？会有多痛苦？例如，注册一家公司需要填写十几张表格，其中有大量重复信息。这虽然痛苦，但为了成功地注册公司，却不得不忍受。
- 如果用户的这些问题不能得到解决，他们会怎样？例如，如果没有一个适合中国用户的Android应用商店，那么用户只能去很多网站上逐个下载需要的应用，或者去学习英文。
- 如果产品或服务不存在，用户使用什么产品或服务去解决这些问题？例如，如果没有《愤怒的小鸟》，用户会去玩《植物大战僵尸》。
- 你的产品或服务与现有的产品和服务最大的不同点是什么？例如，Android的Google联系人可以自动同步到Google账户中，这样可以帮用户节省大量的时间，并且方便集中管理联系人。
- 用户愿意放弃之前的使用习惯而使用你的产品吗？例如，一个Android手机需要2000～3000元，但它提供了随时随地办公的便利，很值得。
- 如果用户愿意使用，产品的定价是多少用户才愿意购买，是免费还是1元钱或1000元钱？例如《植物大战僵尸》，用户最多会付1美元。
- 如果是一个新产品，如何促使用户产生使用这种产品的需求？例如，GPS帮用户节省了很多看地图找路的时间，节省了因为走错路而浪费的汽油，更重要的是，它让驾驶更安全。
- 产品的市场定位是什么？例如，新浪微博定位于中国大城市的白领。
- 市场宣传方式和对象。例如，采用以SEO和SEM为主的方式，客户群是从事IT行业的女性白领。
- 投资回报比是否合理。例如是否可以在3年内收回投资回报。
- 如果竞争对手降低价格，该如何应对？例如，需要注入更多的资金，并且改进增值服务。

上述问题是不可能在产品需求获取之初就完全弄清楚的，要在产品开发的各个阶段不断地“请”用户来验证——进行产品评估和用户访谈。这意味着在产品开发的各个阶段中这些需求都可能发生变化，因此需求管理的模式就需要是一个动态灵活的方式。敏捷方法提供了一个模型，由需求来源、需求池和待开发功能点三部分组成，如图2所示。

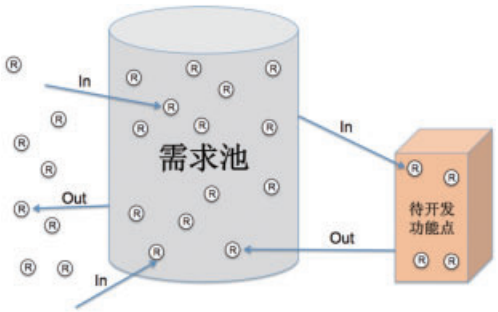


图2　敏捷方法中的需求管理模型

其中“R”代表需求，在需求池和待开发功能点两个池中流动。根据用户、市场和竞争对手的变化，重新评估需求的ROI（投资回报比）和重要性来决定其优先级，从而做到始终开发优先级最高的需求。

当你找到了上述问题的答案，并且通过用户的反馈不断修改和完善这些答案时，你就获取了大量的需求。但这些需求并不能作为直接用于开发的需求描述或者功能点，需要进一步的分析、整理形成需求描述和界面原型（Mockup）。分析方法很多，本文不一一赘述，仅推荐几个简单实用的方法，大家可以在Wikipedia或Google上查询详细信息。

- A Product in Box
- As is ... To be ...
- Customer Insight
- Ideation

- 5W1H
- Five Why

当获取需求和充分分析之后，如何简单地记录和描述这些需求（功能点），更好地帮助一个小的应用团队来管理项目和需求就尤为重要。我推荐使用简化“用户故事”的形式来描述需求。在创建用户故事之初，使用平实的语言将产品的用户场景（User scenario）和用户原型（Persona）描述出来，并在此基础上创建用户故事。以下是一个用户故事的例子。

作为招商银行手机客户端用户，我想要使用手机进行转账，从而节省每次切换到Windows系统所花费的时间。

验收条件为：不需要人工帮助；安全；转账成功或失败要有反馈。

需求描述的粒度要因项目而异，不教条。例如一个iOS平台下的围棋游戏，其中一个需求的描述是“悔棋”。虽然只有两个字，但已足够了，通过这两个字，设计师和开发人员可以面对面地交流，发挥无尽的想象，而不必依赖设计师一个人的经验。

这样的需求描述方法，从用户出发，简单、清晰，可以测试，工作量可以估算，并且可以估算该功能的ROI。

Mockup作为需求描述和评估的重要工具已经被广泛应用。这里推荐几个工具，同时给出界面原型评估时要注意的要点。

推荐的界面原型绘制工具如下。

- Baslamiq：提供包括移动互联网应用和Web应用的原型绘制模版。
- Photoshop或GIMP之类的图形图像绘制工具：对视觉要求很高的应用。
- PowerPoint：提供各种图形和演讲模版。
- YAML Builder：原型可以产生一部分HTML/CSS代码。
- Story board：手绘的界面原型，交互性强，非常灵活。
- HTML/CSS：写代码，尽量模块化，可复用程度要高。

原型评估的注意事项如下。

- 不要让用户设计产品，用户对其他产品的爱憎和意见要重视。
- 不要指导用户使用产品。
- 询问用户使用产品时的感想和思考过程。
- 用户期望下一步会发生什么。

短周期快速迭代，会议高效

解决了需求管理问题之后，接下来就是如何能快速地实现功能，并且尽早让用户来验证产品设计。对于开放平台下的应用开发，建议将一周时间定为一个迭代（迭代是一个固定时间长短的开发周期），迭代的起始时间不要安排在周一或周五。每个迭代要交付一定数量的功能点，原则是确定这些功能点可以比较完整地解决用户的一个或几个问题。每个迭代的第一天发布上个迭代通过验收的功能点。

使用持续集成保证代码尽早集成并能轻松发布。对关键且稳定的功能使用自动化的用户验收测试保障。可以使用Selenium、Cucumber之类的测试框架。

开发过程中往往会有固定和非固定会议。有时，这些会议占用了大量的时间而没有起到团队期望的效果。针对开放平台的应用开发，建议采用以下会议形式及其时长。

- 站立会议：每天1～2次，沟通项目进度及遇到的棘手问题，每次5分钟以内。
- 迭代计划或设计讨论会议：每个迭代，每次1小时以内。
- 敏捷回顾会议：选用，每1～2个迭代一次，每次1小时以内。
- 对开发人员来说，其他时间应该用于设计、开发和质量保证。每天保证6小时的有效编码时间。

工程实践的灵活运用

要实现快速迭代、频繁发布需要最佳的软件工程实践的保障。这些实践包括自动化单元测试、持续集成、版本管理、稳步重构、测试驱动设计和

结对编程，等等。这些实践需要根据团队的经验及其技术能力有选择地应用，不能强求。尽量让团队自主选择适合自己的实践而不要自上而下地强加。

软件开发离不开高效的工具支撑

“轻敏捷”方法需要高效的工具支撑，一些创业团队通常喜欢选择相对比较成熟且开源的产品和工具。推荐的“轻敏捷”工具集如下。

- GIT、SVN
- CruiseControl、Hudson、Jenkins、Selenium和Cucumber
- MediaWiki
- Baslamiq、Axure、UIKit、YAML Builder和ClickTale
- 白板、故事卡、便签和Trac
- IDE
- Google Analytics、A/B Testing（Google Optimizer）和Google Webmaster
- Nagios、Cacti
- Pentaho、MapReduce、Hive、AWS EC2和Rackspace

开放心态，真诚反馈，拥抱变化

由于开放平台下开发的特殊性——需求的不确定性强、可变性强，所以对于开发团队来说，“开放心态，真诚反馈，拥抱变化”的文化是保证产品和团队持续完善的制胜之道。只有善于吸纳、分析、提炼信息，且善于学习，才能促进产品和团队的成长。如果团队中的每个成员都有自己独立的见解，那么出现不同意见是再正常不过的事情了。因此，很多创业团队在沟通交流上遇到了巨大的困难。每个人都想坚持己见、站在自身立场去思考问题，而不愿意倾听，那么无论是给予还是接收反馈都会带有负面的情绪。在这样的团队文化下，很难开发出成功的产品，也很难提供用户满意的服务。给予和接收反馈要遵循以下几条原则。

- Treat everyone as treat yourself（对待任何人要和对待自己一样）。
- 站在接收方的立场，给予真诚的反馈。
- 反馈是为了增强彼此的信心，并且弥补不足。

中国IT行业有一句名言：“杀死一个程序员不需要枪，只要变三次需求就好了。”开放平台下的应用开发需求会变化更频繁、更不可预见。在这样的客观现实下，团队需要学会合理地面对产品设计的变化和产品方向的变化等。以客户为中心，重视市场反馈。

这个世界上没有包治百病的银弹和放之四海而皆准的方法。团队负责人需要在掌握一些基本的方法论的前提下，带领团队去摸索适合自己的方法。当然，这个过程开始或许会痛苦，也不可能一蹴而就，如果能得到导师和专家的指导会少走很多弯路。

“轻敏捷”特点

在本文结尾，我们总结一下轻敏捷的特点。

- 更少文档。
- 更多面对面交流。
- 更少重复工作。
- 更多自动化（DevOps），更多工具。
- 更少闭门造车。
- 更多快速迭代和快速验证。
- 更少的耦合。
- 更多的各角色做自己最擅长的事，DSL。



王晓明

知名组织转型和敏捷专家，高级管理顾问，“阿拉丁”敏捷组织转型框架创始人，敏思特咨询首席合伙人，创新工场特聘导师，腾讯高级管理顾问。新浪微博：@王晓明同学。

责任编辑：杨爽（yangshuang@csdn.net）

如何设计高转化率的网站

文 / 陈世欣

网站转化率的高低直接影响网站的收益，因此，如何设计高转化率网站成为事关网站成败的重要一环。本文分析了在流量既定的情况下，影响网站转化率的因素，同时指出了设计高转化率网站的要点。

“网站存在的意义即在于转化。因为每一个网站的存在都有它的目的，而这个目的正是需要的转化结果。”

例如电子商务网站、门户网站、垂直网站及政府网站等，每类网站都有它们清晰的诉求和目标，因此需要从访客到目标的明确转化。

转化率越高，网站越好。例如，电子商务网站A的每日IP高达10万，但每天成交只有200单，网站转化率只有2‰；电子商务网站B的每日IP可能只有1万，但每天成交也有200单，网站转化率有2%。收入相同但投入成本却相差不止10倍，竞争力显然差异很大。

在流量既定的情况下，影响转化率的因素直接影响转化实现的程度。重视和研究网站转化率，有助于针对性地分析网站在哪些方面做得不足，哪些广告投放效果比较好，可以迅速提升用户体验、节约广告成本。

影响网站转化率的因素

从根本上说，影响网站实现最终转化的两个关键点是：正确的访客和正确的访问体验。也就是说，网站要能吸引真正对网站内容和服务感兴趣的人，且当这些人访问网站时，能获得良好的用户体验以促使转化。这二者缺一不可。归根结底，对转化的优化就是对这二者的不断优化。

正确的访问体验取决于三个主要因素，可用图1中的转化三角形描述。

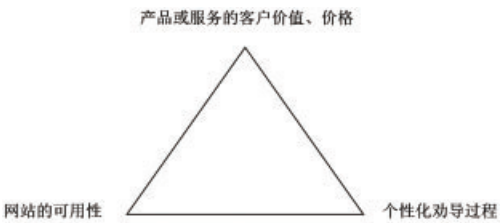


图1 影响访问体验的三个主要因素

产品或服务的客户价值越高、价格越低，则对访客的吸引力越大，也就更容易转化。例如，如果商品和服务大幅度折价出售，或者赠送礼品等，能够大幅提升转化率。

网站的可用性越好，越易用，障碍越少，也就越容易转化更多的客户。

考虑到客户的心理和知识背景不同，个性化劝导过程能根据客户的实际情况，把握用户的心理，打消客户的顾虑，提供适当的信息及逻辑，重点描述产品或服务的最吸引客户的特性和价值，让客户服从引导，从而改变客户的态度或行为。

在实际的网站运营中，有很多细节可以影响转化率的三大主要因素，且企业里很多人的决策都会对转化率有影响，包括产品设计、网站管理、网站营销等各个岗位。网站（特别是独立网页、着陆页面）定稿的权限常常分散在很多人手里，很多时候，网站（网页）该这样还是该那样的决策，常常由非专业人士（并非专业的用户体验设计

师）来做。比如，网络营销中有很大一部分工作是在做网站优化，以提高转化率。就算有个好的用户体验团队，也只能间接影响设计。更何况很多公司的网站设计师不靠谱，设计过程也未必合理，这样在沟通和配合上出错的概率非常高。

设计高转化率网站的要点

要设计高转化率的网站，需要做到以下三点。

第一，鼓励团队工作人员在同一个转化率优化的思考框架中进行思考和沟通。

为了便于设计师、产品经理、营销人员等岗位的人都能充分理解和分析网站转化率的各个因素，关注影响转化率的关键因素，做出正确的决策并密切配合，就需要大家在一个转化率优化框架下沟通。这样可以让思考更全面，强制性地把所有因素都罗列一遍，防止遗漏。迫使设计者从每个角度都至少想一想，帮助设计者发挥出他本身应该有的能力。

团队头脑风暴时常常容易跑题，同一个思考框架可以使每个阶段的话题更集中，统一用词之后也能方便沟通和理解。

框架便于知识分享和能力训练。培养新人时，零零碎碎教的东西很难被接收，有一个框架会更有利于理解和思考。在工作时，这个框架也有助于新人更快地掌握技能。

第二，根据网站的目标，利用可用的劝导元素和资源，构建劝导系统的策略和步骤。

劝导系统是对网站存在的劝导元素（信息、社会化元素和各种证据）实施规划、设计、开发和优化的一种方法。这种方法确保网站优化工作始于一个坚实的基础，能更有效地劝导网站访客（待劝导对象），并让网站优化者容易衡量具体工作，更明智地运用劝导实体。

优秀的劝导系统应该做到：向真正需要的人出售产品或服务；让访客自愿点击网站，进而吸引、劝导访客，避免出现任何带有强迫性的标志；对访客可能提出的问题、存在的疑虑、注意事项有预先准备；强调消费可以使访客受益，不要只强调产品的特性。

设计劝导系统的四大核心关键点是：建立高信任度场景；提供社会化的证据证明其他人已经信任；对不同客户群讲有针对性的故事，打动人心；强调紧迫性（时间有限，库存有限）。

第三，提出一个科学的基于数据分析的设计和优化工作流程，用数据分析每一个微观转化，使用科学的方法和流程测试筛选正确的选择。

网站设计者希望来到网站的每名访客都能够完成一个宏观行为（网站的最终目标），但其中必然需要通过很多个中间步骤的“微观”行为。每个步骤都是转换率的一部分，客户需要经历一系列的微观行为的转换。每个转换点必须能成功地劝导访客进行下一个步骤。每个微观行为就像是螺旋线上的拐点，因此关系重大。当设计者把漫游路径上的每个步骤都当做可检验、测量和优化的转换点时，就会真正了解是否成功地在每个阶段促使访客进行转换。

总结

综上所述，要提升网站的转化率，就要不断提高产品的客户价值，不断提升网站的可用性，改进用户体验，不断改进对客户的劝导过程。除了熟知的用户体验之外，还要花更多时间关注对用户的劝导过程，设计合理的劝导系统。

当然，由于网站所处行业不同、客户群不同，所采取的方式方法也有所不同，这就需要不断去研究行业，揭示过程，发现问题，用A/B测试或者多变量测试等方法试验解决方案，用数据驱动网站的改进。



陈世欣

欧洲2Style4You公司首席技术官，负责公司的产品和技术，曾任易趣网项目经理、安家网副总经理、美国Movoto.Com中国公司总经理、和家网首席技术官。

责任编辑：杨爽（yangshuang@csdn.net）

听市场需求，而非个人需求

文 / Jeff Lash　　译 / 李勇



产品经理的工作看似简单——发现产品需求、开发产品、赢得客户。可接下来，就没那么顺利了：客户要求升级产品特性；销售部门为了让产品好卖，将产品的必备特性列了长长的一串；竞争对手开始山寨你的创意，还用新增的产品特性骗走你的客户群；高管们七嘴八舌提出些“绝妙”的点子，企图在下一代的产品中付诸实践。

产品经理几乎每天都要应对增加或修改产品特性的要求，却无法一一满足。这些令人抓狂的要求来势汹汹，任何产品经理都招架不住。

相信大家都曾被这样的指令洗过脑——“要以客户为中心”。大家都坚信客户提出的每个要求都是理性的，违反客户的“圣旨”就是“死罪”。然而这样的观点只会让事态更糟。产品经理要取得成功，就不能以单个客户需求为标杆，应将市场的整体需求作为考察对象。

如果把赢得客户的过程看做一场选举，那么现有客户就是你最重要的选民，通过投票，我们能很容易地发现他们喜欢什么、不喜欢什么。实际上，有时想屏蔽他们的意见都难。但你必须意识到他们已经投了你一票，而潜在的客户远比现有客户多。如果产品一味讨好现有客户，怎么可能让业绩不断增长？如果不考虑潜在客户的需求，又怎能拓展新市场？

产品经理与那些没有购买我们产品或者弃用我们产品的人交流，能学到更多的东西。如果你问现有客户“不喜欢产品的哪些地方”，他们会列出一堆希望添加或升级的产品功能，而这些功能往往是他们购买产品之后才想到的，并期望免费添加进来。而竞争对手的客户则会告诉你他们为什么选择别人的产品，这时你才能知道该怎样去完善自己的产品。与全新市场中的客户交谈，他们会告诉你自己现有的问题及为了解决这些问题，他们愿意花多少钱。这才是真正有价值的信息。

一旦将市场作为一个整体来考察，开发出真正具有价值的产品指日可待。与其为了个别用户的需求对产品修修补补，不如开发新的业绩增长点，赢得更多客户的青睐。

现有用户能告诉你今天遇到了哪些困难，却不可能告诉你三年之后会陷入何种窘境；不可能告诉你其他行业所面临的问题；不可能预测未来技术的进步；更不可能告诉你为什么其他人没有选择你的产品。可是，现有客户的要求也不能被忽视，特别是在当前业绩增长完全依靠现有客户的情况下（例如，定制产品、SaaS软件运营服务）。但一定要记住，将市场的整体需求作为考察对象。

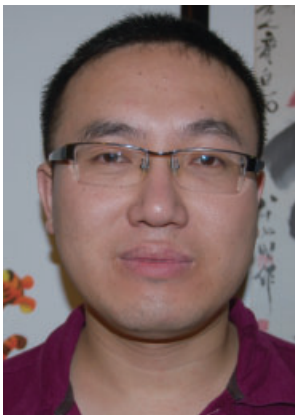


本文节选自《42 Rules of Product Management》，作者为Brian Lawley、Greg Cohen等人，感谢Happy About出版社对节选内容的中文授权。
本书主页为 http://bit.ly/JYmGlo。

一分钟先生

如何辅导新人

有效地辅导新人，使其尽快适应岗位要求，胜任工作，不仅可以节省大量的培训成本，更能确保企业的健康发展。本期三位嘉宾将结合自己多年的实践经验，和大家分享辅导新人的技巧和方法。



李强（在宽）
天猫产品技术部资深经理

心系新员工，重视辅导工作

我们对新人的辅导在入职前就已经开始了。

- 营造相互分享、相互成就的团队文化。在条件容许的情况下，尽量安排团队面试。帮助新人不仅是主管一个人的事情，也是团队所有成员的事情。
- 平时做好流程梳理和知识沉淀工作，确保研发流程、文档模板、编码规范和各种最佳案例等内容全面清晰，团队所负责产品和系统的文档及时更新，制定全面的新人手册。
- 做好“导师”的选拔和培养工作，有个好的导师等于新人成功的一半。团队可以定期组织导师经验交流会，促进导师自身的成长。

新人入职后辅导要具有针对性，因人而异，区别对待。

- 对于新入职的毕业生，要着重培养他们的工作方法和工作技能；要培养他们的好奇心，能够从看似简单的日常工作中追本溯源，探寻问题背后的本质。例如，是否清楚流程背后的出发点，是否清楚所调用模块的工作原理。此外，还要帮助他们树立自信，帮助他们设定一个合理的目标。
- 有工作经验的新人一般有自己的工作方法，工作技能也基本能满足岗位需求，主管需要着重在工作态度和团队融入方面做好辅导工作。要为这些新人，特别是资深人士，一个相对宽松的试用空间，让他们不急于证明自己，而是多去倾听、多去了解和接纳现在的团队和产品。
- 因人而异地安排好第一项任务，因为新人的第一个项目对他们的成长和融入团队至关重要。新人在第一个项目中表现的好坏，会直接影响到他们对自己在新环境的预期，也会影响到团队其他成员对新人的信任。因此，给新人的任务宜小而多，不宜大而少。小而多的任务，主管容易跟进和调整，新人也便于掌握工作进度，同时随着一项项小任务的完成，新人也会越来越自信。
- 及时安排新人参加公司文化方面的培训。这些培训一般包括企业组织架构、管理制度、企业文化、品牌理念等，还要组织团队建设方面的活动。在这样的跨团队培训中能认识很多不同部门、不同岗位的同事，为日后展开工作提供方便。

我们往往能从一个团队的共性看到团队主管是否对新人进行了有效的辅导。在新人身上花费一分力，通常能得到三分回报。归根结底，要做好新人辅导，辅导者本身要有爱心，心里有新人，才能真正帮助到新人。



陈宗绵
开发工程师

制定不同的辅导方法

技术部门的新人一般分为两类：一类是没有开发经验的，另一类是有一定开发经验的。本文将结合这两类新人的特点，讲述相应的辅导方法，抛砖引玉，帮助大家找到适合自己公司文化的辅导方法。

没有开发经验的新员工

对没有开发工作经验的新员工，我会在较长时间（一般为半年左右）里把他们限制在现有的开发规范下，重点培养他们的开发习惯和开发思维，具体有以下几个方面。

入职培训。通过入职培训，让新员工了解团队的代码开发规范及整个开发流程。每家公司的开发规范和开发流程不同，并不能用简单的好或坏来衡量。这些只是为了让项目本身在代码风格上具有较强的一致性，方便阅读和维护。

代码检查。严格要求新人遵守现有的开发规范和开发流程。在新员工入职后的半年内，我都会在他们的开发过程中随时检查他们的代码，以便及时发现问题，立即纠正错误、处处留心破窗效应和蝴蝶效应。

开发手册和自检清单。开发手册以目前项目的代码框架、开发规范为基础，详细地告诉新员工如何进行一些功能开发，提供示例代码。简单到如何写一个按钮，细致到涵盖日常开发中需要的每个功能。自检清单以目前项目的开发流程为基础，内容包括但不限于以下几项：代码签入SVN时是否写上注释，是否对自己开发的功能和相关的整个业务流程进行自测等。严格要求他们对照自检清单上的每一项认真检查自己的工作完成情况。

及时沟通。告知新员工遇到无法解决的问题，要立即请教其他同事，不要自己过分钻研。如果在15分钟内无法解决某一问题的话，就要马上请教。

帮助新员工进步。帮助新员工从技术上进步。例如，提供一些技术图书和网站、定期组织内部培训等。鼓励在阅读到好的技术文章时积极在同事间进行分享，必要时要进行一定的讲解，营造良好的学习氛围。

有一定开发经验的新员工

辅导有一定开发经验的新员工，主要体现在开发思路上的引导，如下所述。

允许试错。在项目允许的情况下，提供各种试错机会。让他们从自己的失败经验中总结经验。说服教育对处在这个阶段的开发人员来说不再那么有效。

互相交流，达成共识。定期开展评审和头脑风暴等活动，让他们产生共识，增加团队员工之间的默契。处在这个阶段的开发人员对别人的处理逻辑会存在各种不同的意见，但在一定程度上讲，不同的处理方法对功能的实现效果和代码执行效率影响不会太大。如果缺乏共识，那么就需要多加沟通。

全方位发展。可尝试带着新员工学习各种技术。不管是前端的HTML和JavaScript，还是后端开发的PHP和Java，以及各种安全问题和算法问题。拓展新员工的知识面，以便让他们知道某一环境的特定业务场景下需要用什么技术手段来解决业务问题。

业务能力培养。结合实际情况，允许开发人员离开技术岗位，到业务岗位去了解和学习业务。只有这样，他们才能熟悉业务场景，不会闭门造车。

善于总结，提高效率。帮助开发人员进行各种工作总结，找出重复的工作，并指导他们自己开发一些代码生成器之类的小工具，来提高日常工作效率。

建立合理的新员工培养制度



段念
乐元素游戏开发公司技术副总裁

健康发展的公司通常需要合理的人员流动。进入公司的新员工能够给公司带来新鲜血液，为公司的发展带来新气象。新员工有两类：一类是在其他公司有一定经验的人员；另一类是刚毕业的学生。

毕业生有激情却容易眼高手低

作为刚走入职场的新人，应届毕业生一般都会怀揣着对未来生活的美好向往和强烈的工作热情，朝气蓬勃、干劲儿十足。但同时，由于缺乏社会经验，刚毕业的学生对事物的认知往往过于理想化，朝气蓬勃却很容易遭遇挫折；干劲儿十足有时却会过于鲁莽；有激情却容易眼高手低。

虽然在学校已经系统地学习了相关的技术，但他们却缺乏对企业开发的认知和实践经验，通常要经过一段时间的培养和实践才能达到企业在技能方面的要求。很多大型软件企业早已建立了完善的员工培训体系，而且有足够的资源来应对工作任务，因此可以花费较长的时间来培养新员工。但对大多数资源并不充裕的软件企业来说，让新成员快速成长、快速发挥作用是一个重要的课题。

制定合理的培养制度

德雷福斯学习模型指出，有效的学习环境应该有以下特点。

- 学习者有明确且具体的任务，而且该任务要有一定的挑战性。
- 学习者需要有一个可以重复实践和犯错的环境。
- 学习者能够及时获得反馈。

我们对新员工的培养大体上基于德雷福斯的学习模型，从制度、项目和人员三个方面为职场新人提供高效的学习环境。

在制度方面，我们公司采取的制度是“导师负责制”。我们会为每一名新入职的应届毕业生指定一位导师，负责新员工的成长。同时，为了保证导师能够有充足的时间和精力辅导新员工，规定每位工程师最多只能带两名新员工。这样，每名新员工都能快速熟悉开发环境，快速了解项目组的开发方式。导师一般由项目经验丰富且具有良好传授者潜质的工程师担任。考核时，会把导师的工作效果与升迁和奖励关联起来。

在项目方面，在每名新员工入职的第一天，我们就会为其分配具体的项目。在项目中培养人员是我们的惯例，因为只有这样才能让毕业生有机会实践和提高。导师和项目负责人会共同为新员工布置任务，布置任务和检查任务的原则如下。

- 保证所分配的每项任务都有明确的完成目标，且有一定的挑战性。
- 分配的任务不能是项目中至关重要的或有极强时间要求的任务，保证新人有犯错的机会。
- 以天为单位检查目标达成情况，每天及时反馈意见。

在人员方面，由于每个人的性格和认知不同，所以在基于制度和基本原则的前提下，导师还需要根据每个新人的性格特点给予稍微不同的对待。例如，对MBTI性格倾向性为“内倾”的员工来说，过于频繁和密集的工作进度检查会让其觉得不适应；而对于计划性不是很强的员工来说，则需要和他共同制订较为详细的计划并仔细检查其计划完成情况。

提升和培养新员工的投入量的确不少，但如果能够在组织内建立合理的新员工培养制度，一方面可以使组织的生产率和效率大大提高；另一方面可以让新员工很好地保持精力和干劲，发挥超出预期的作用。对每个软件企业来说，这个投入是非常值得的。

移动产品用户体验与设计系列（一）

做正确的加法

文 / 林敏

最近时不时就会听到关于做减法的提醒：生活要做减法，战略要做减法，产品要做减法，创意要做减法……一时间仿佛什么都在做减法。我曾参加过一个移动互联网创业的活动，注意到几乎每个创始人在介绍产品时都会有意强调自己在做减法，似乎没有了减法产品便不够吸引人了。做减法本身无可厚非，它对于避免软件膨胀的作用是显而易见的。但让我感到担忧的是，当下对减法的追捧开始冒出信仰的味道，从而许多人不加思索地就把加法摆在对立面，避之唯恐不及。

其实加法也是重要的产品创新手段。鸡尾酒是加法的典型例子，而最具影响力的加法式设计大概要算瑞士军刀了，它受欢迎的程度并不亚于苹果公司的产品。

加法和减法本身并没有对错，会犯错的都是用这些方法的人。

正确地运用加法反而容易产生出奇制胜的效果。三星在2011年10月推出的Galaxy Note智能手机，短短5个月内就取得了500万部的销售成绩。我还看到某手机网的主编在感慨想不通为什么Galaxy Note会火。其实答案很简单，它不过是做了两个简单的加法：笔和大屏。乔布斯对于触摸屏手机有一句著名的论断：“上帝给了我们十支手写笔，我们不要再多发明一个了。”于是设计师们谈笔色变。我曾参与一个用户调研，其中Galaxy Note的用户在被询问他们选择这款手机的原因时，大多数用户不约而同地都会提到笔是影响他们购买决策的一个重要因素。同样，当设计师们都痴迷于iPhone的3.5寸屏是单手触摸操作的极限选择时，他们选择了无视屏幕大小与信息呈现量的关系。Galaxy Note的5.3寸屏不过是通过增大信息呈现量而让对此敏感的用户感受到质的差异。顺带值得一提的是，Galaxy Note的大屏也是为了更好满足笔的使用而自然衍生的设计思考。

因此，正确的加法同样能够带来成功的产品。我们应该把焦点落在如何正确地使用这些方法上。正确的加法必须具备以下两个特点。

- **第一，它必须是核心体验的合理延伸**。橡皮之对于铅笔的加法，是不是自然到当我们看到时丝毫不会产生任何突兀感？这就是从书写这个核心体验延伸出用于修改书写错误的橡皮之设计合理性。

- **第二，它必须能够在很短时间里被用户理解和感知**。选择鸡尾酒对很多人来说不是件容易的事。虽然酒单上写着每款酒是由哪些酒和哪些辅料调制而成，但只有饮在嘴里才能让人真正感受到它的魅力。但鸡尾酒的这种新体验是可以非常快速地被感知的。电子产品的数字罗列本质上是鸡尾酒模式，但恰恰缺失了快速感知的部分。这也正是现如今电子产品体验店盛行的重要原因。

减法是为了专注，而正确的加法会带来欢乐。“六一”儿童节被苹果App Store推荐的应用《鲸鱼岛的冬天》就非常好地体现了这一点。它将游戏首页加上模糊处理，使之看起来像是玻璃上结了一层水雾。小朋友们会很自然地用手去擦以获得清晰的页面。据说许多小朋友因为这个有趣的加法而仔仔细细地把整个页面都擦干净以后才开始游戏。产品设计，就是要这般通过细微处带来的欢乐感动用户。那么，现在就把加法放回你的工具箱吧。



林敏

用户体验践行者和布道者，曾创立UU网为国内早期从业人员提供学习交流平台。现为三星中国设计研究所用户体验创新部负责人。

责任编辑：陈博（chenbo@csdn.net）

Windows 8能否推动移动互联网新变革？

文 / 马宁

本文从移动设备、Metro UI、编程的变化、Windows Store四个方面分析了Windows 8在移动互联网时代带来的影响与面对的挑战。

Windows 8 Release Preview版在2012年的儿童节正式发布。虽然不如外界期望的那么成熟，但Windows 8已开始慢慢地学步了。作为开发者，我们面临的挑战要大于之前Windows的每一次升级，可能仅次于当年从DOS升到Windows。别忘了，无数在DOS下非常成功的软件因为无法支持Windows而在一夜间烟消云散。

移动设备

Windows 8的设计原则从一开始就被设定为“平板优先”，甚至为此牺牲掉了一些使用键盘、鼠标时的操作体验。其实Windows 7已达到了PC操作系统的巅峰，而Windows 8的很多颠覆性设计都是因为微软要抢夺平板电脑市场。微软之所以如此看重平板市场，一方面，是因为微软在长达十年的时间里没有革命性的产品推出，一直在吃PC的老本，但PC领域已失去了快速发展的动力；另一方面，借由iPhone和iPad的成功，苹果的电脑和笔记本严重威胁了微软在PC市场上的统治地位。

微软早在十年前就制定了自己的平板战略，推出了Tablet PC，就像早夭的Newton一样，Tablet PC也没有获得市场的欢迎。一方面，是因为Tablet PC的硬件由电脑厂商设计，它们坚持认为要为Tablet PC增加键盘，这增加了成本和重量，使Tablet PC无法单手操作；另一方面，Tablet PC不是一个全新设计的操作系统，只是增加了触摸屏，UI方面仍延续了传统Windows的用户体验。

可以说，Windows 8总结了Tablet PC的失败和iPad的成功，重新设计了用户体验，使其能够符合平板电脑的需求。

而硬件的发展也使得Windows 8的成功充满了机会，Intel希望借助低功耗CPU杀回移动设备市场，况且Windows 8 RT也会支持ARM CPU，而无数PC厂商也希望杀入平板市场。平板产业链在经历了上网本、Android平板的轮番轰炸后，Windows 8也成为平板市场的下一个希望。

不过，其中也存在很多隐忧，例如微软对于硬件设计方面没有太多控制力，每家平板厂商都想尽办法创造与众不同的平板产品，但在软件和硬件无法密切配合的情况下，能够做出匹敌iPad的产品，可能性恐怕不大。如果用户体验方面无法匹敌，那么就只能在价格上想办法了。目前还不清楚Windows 8授权费用的具体金额，但如果太高的话，恐怕无法吸引硬件厂商。

微软不会直接授权中小企业来生产Windows 8平板，但由于软件和硬件的通用性，山寨版平板估计会在三个月内上市，半年内就可以将Windows 8平板的价格压缩到一个不可思议的程度。这样会让整个产业链快速失去盈利能力。

Metro的革命

Metro UI已成为了设计领域的一股新鲜空气，发



Metro UI的一大设计原则就是内容重于形式

源自Zune，成熟于Windows Phone的Metro UI，最初的设计灵感是来自地铁系统的指示牌。Metro UI的一大设计原则就是内容重于形式，我们可以看到超大的字体、简单至极的、充满2D色块的图标。经历了两年多时间，Metro UI已逐渐被用户所接受，形成了自己独特的风格。

但Windows 8中的Metro UI有自己的独特性和设计原则，每个应用程序的主界面上应该分组显示主要的信息和内容，点击进入详细界面后，应该有后退按钮。

信息应该是流动的，而且横向滚动成为主要的交互方式。这一点让很多开发者无所适从。当然，Windows 8也增加了很多有趣的功能，例如用户界面可以贴靠在屏幕的一侧，这对聊天类的软件来说非常有用。

新增的Contract功能，允许应用程序将内容分享到不同的应用程序中，而且也可以将自己的内容加入到搜索功能中。

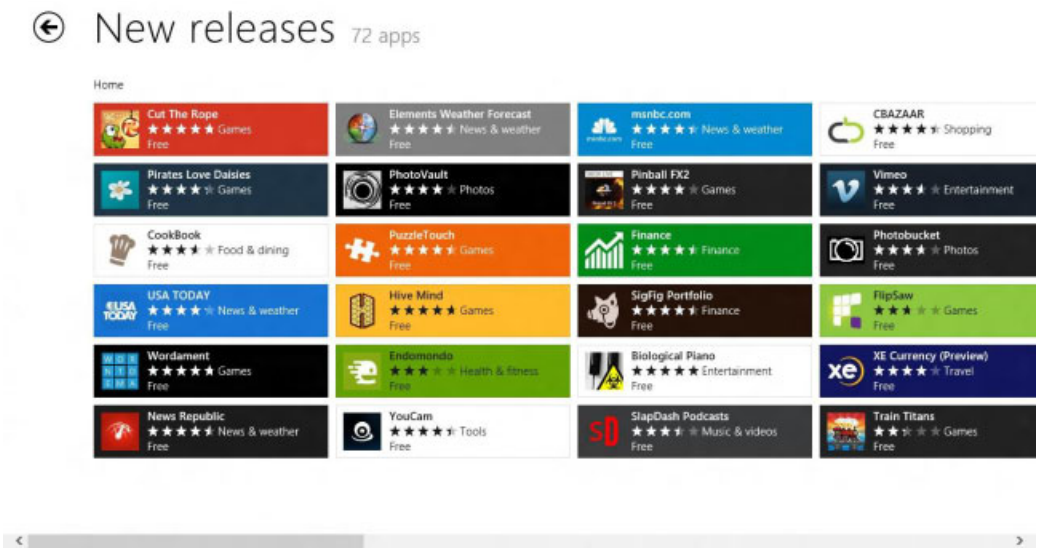
但Windows 8最大的困扰是如何将传统的桌面应用和Metro应用整合在一起。毕竟Windows 8还要服务于使用键盘和鼠标的PC。但Windows 8在PC模式下的很多交互设计细节还没有得到完善。在鼠标控制时，从屏幕边缘擦出控制界面的动作变得十分困难。如果Windows 8没有办法很好地服务于PC用户，那么Windows 7可能还会像Windows XP那样服务很多年。

编程的变化

这次Windows 8的挑战主要来自Metro UI，适合移动设备多点触摸输入方式的UI Framework。而Metro UI的编程方式类似于Silverlight，结合了Windows Phone中的一些特点，但Metro Style UI Framework却和两者都不相同。

Metro UI支持三种编程语言：.NET（C#、VB.NET）、C++和JavaScript。在这些编程语言的下面，是统一的WinRT Framework。Metro Style应用程序对于Windows Phone开发者来说并不陌生。不过，Windows Phone 7上的Metro UI只能使用Silverlight技术来实现。Windows 8上的Metro Style应用程序则能够支持C#/VB.NET、C++和JavaScript三种方式，而且底层的API也从Win32变成了WinRT。

自从Windows 2000开始，Win32 API几乎伴随了我们这一代程序员的整个职业生涯，MFC时代熟练使用Win32 API是高手的标志。后来有了.NET，使用P/Invoke调用Win32 API也是我们解决某些棘手问题的不二法门。但WinRT取代了Win32之后，我们再也不能模仿鼠标输入事件了。WinRT是一个Native的API库，所以C++可以进行调用，而C#/VB.NET则是通过.NET封装来调用的，界面描述则是使用XAML，最大的挑战则是来自JavaScript调用。微软使用JavaScript的语法来描



Windows Store的模式将使物流成本趋近于零

述了Metro Style UI，但底层类库的调用仍旧是基于WinRT的。因此对于传统的JavaScript开发者来说，仍需要学习WinRT类库的调用方式。

另一个挑战是，使用JavaScript编写的Metro Style应用，代码是无法复用到普通的JavaScript应用中的。这不符合开发者的利益。因此，在很长一段时间内，Windows 8上最活跃的开发者社区仍旧是.NET，因为C++开发者要学习XAML，而JavaScript开发者要学习WinRT。

在编程模型上，Windows 8向Windows Phone 7借鉴了很多有益的地方，例如所有应用程序都是运行在沙箱中，有自己的独立存储空间，后台运行时进行严格限制，从这些方面来保证操作系统的安全和性能。当然，Windows 8也修正了Windows Phone 7中的一些问题，例如无法进行程序间的相互调用。Windows 8是使用Contract来解决这个问题的，Contract的机制类似于Android中的Service，为程序间的调用提供一种标准接口。首先，由被调用者声明Contract，然后由调用者进行相应的调用。应用程序可以声明一个由Search或Setting调用的Contract，在搜索系统功能中，就可以加入自己的搜索结果。

Windows Store

Windows 8另外一大卖点是Windows Store的引入。能够将软件送达每一台PC的桌面上，恐怕是每一个ISV的梦想吧。Windows Store的游戏规则与Windows Phone Marketplace类似，采用7∶3分成，超过2.5万美元的，则按照8∶2进行分成。

不过，Windows 8面临的问题比iPhone要复杂。最初的压力会来自原有的软件分销体系，Windows Store的模式将使物流成本趋近于零，而软件的数量趋近于无穷，这种典型的长尾理论应用会造成之前分销体系的崩溃，但分销体系的压力也许能够迫使微软作出一些妥协。Windows Store目前已经支持传统桌面应用程序上线了，但Windows 8仍旧允许用户自行部署应用程序，而这种方式会对Windows Store上的软件形成非常大的反盗版压力。不知道微软如何平衡这两种新旧体系共存的问题。

Windows 8真正吸引用户的杀手级应用是Office，几乎所有iPad用户都承认iPad在办公方面的性能是不如娱乐方面的。而三星的Windows 8平板增加一个底座和无线键盘就可以成为便携式工作站。目前，基于Metro UI的Office还没有露面，让我们拭目以待吧。



马宁

北京支点联游CTO，OpenXLive开发者。曾任微软亚洲工程院软件工程师，从事Windows Embedded Compact开发工作。加入微软之前，连续四年获得“微软最有价值专家”称号。

责任编辑：陈博（chenbo@csdn.net）

游戏画面的印象价值和审美属性分析

文 / 郑金条

对于玩家而言，视觉和声效仍是每个人接触游戏时可能获得初次愉悦反馈的最直接方式。作者从画面对玩家初次选择的影响、美术和UI布局、画面架构的协调度三部分阐述了游戏画面的重要性。

Quora上有个问题：对于一款游戏而言，谁扮演了更重要的角色（假定的三个选项为游戏设计师、美工和程序员）。在几个有针对性的回复中，游戏设计师（Alexander Galasso、Pano Anthos和Paul Martin）、程序员（Anatolie Gavriliuc、Michael Hamlin）被认为具有主导游戏的影响，而美工只是在整体游戏的呈现配合中具有价值。

尽管整个开发群体都能够认同游戏的画面和声效在玩家深入接触游戏前是游戏最具直观性的两个导向窗口，但随着玩家对游戏的深入，前端运行的流畅度和后端数据服务的支撑潜力往往成为更重要的评判属性。至于游戏中的故事架构、关卡的设置、数值均衡性以及相关规则与机制的设定早已成为一种基础性渗透，以致作为最直观呈现的载体的美术成分（包括原画、角色、动画和各种UI界面）似乎并未能获得等量的认可权重。独立开发者Martin Pichlmair在2011年时就针对自己的单款游戏研发投入情况进行了解剖，所得的数据包括Code拥有33%最大的权重，Art以25%的比例位居第二，Game Design以16%排名第三（其他还包括占11%的Materials、占10%的Marketing和5%的Production）。

这一点可以从薪酬上得到最为直观的反映，按照*Game Developer*杂志连续两年的数据（2010年和2011年），美术相关从业人员的平均薪酬都低于游戏行业的平均薪酬，2011年游戏产业平均薪酬为81129美元，相应的美术相关从业者平均薪酬仅为75780美元（两者差距在5000美元左右，当然相比2010年时的将近9000美元的差距已经稍微好转）。

画面对玩家初次选择的影响

无论美术成分在游戏研发进程中所占的比重或者话语权为何，对玩家而言，视觉和声效仍是每个人接触游戏时可能获得初次愉悦反馈的最直接方式（例如游戏的曲子让人陶醉或者刚好画面的风格让人着迷），也是开发者在未能完全展示游戏效能前俘获玩家的关键要素。Zynga首席执行官Mark Pincus就认为开发者需要在玩家三次点击内达到吸引玩家的目的，这其中就包括从画面角度渗透出来的游戏品味，Concept Art House首席执行官James Zhang在分析社交游戏画面效能时同样认为画风和页面质量往往能够直接引导用户的印象价值判断，特别是在玩家面临无数的游戏选择而精力明显不够用的情况下，依靠美术的首

次接触导向的印象好感就成为游戏初期留存率的提升方式。

事实上，作为用户审美布局的一部分，画面同样赋予了游戏某种别致的生命力，最为典型的就是Chair Entertainment旗下的《Infinity Blade》，因为玩法单一饱受诟病但其画面效果惊人，同样成为了用户的首选，营收额在数千万美元（制作成本200万美元左右），其续集《Infinity Blade 2》更是在发行的第一个月就获得超过500万美元的收益（无与伦比的画面效果对这款视觉游戏而言就显得独一无二，并以此征服了边抱怨边沉迷的玩家）；而Zynga的Ville系列游戏已使其拥有大眼睛的可爱卡通角色成了公司的游戏形象标志（一旦形成印记，玩家一般就会通过图像来识别游戏，例如更改了马里奥系列游戏的色调肯定就会让熟悉的玩家产生诸多不适应）。

如果这种视觉需求放置在App Store和Google Play的商店辨识环境，就会成为用户是否选择下载的评判依据，并且其效果被放大化。无论是游戏的Icon还是效果和玩法的截图，都迫切需要开发者在有效的画面拼接中对游戏进行展示，并以独特的风格、色调以及布局去争取可能感兴趣但尚未最终下决定的玩家（这个在App的强市场竞争中已经被逐步精化，不够直击游戏核心的展示都会被用户在快速浏览中忽略）。

这种倾向性在玩家的选择中很容易被释放出来，最为典型的案例是游戏机制相近的《Bejeweled》和《Blood War》在玩家的接受度和游戏受众拓展上产生天壤之别的差异。《Bejeweled》作为PopCap最为经典的游戏之一，无论在任何载体上都能够获得不同群体的玩家赞誉，而同样以连线消除面貌呈现的《Blood War》在休闲游戏市场所延续的却是《Chronicles of Riddick》以骷髅形象为表征的血腥氛围（灰暗以及血红、暗色调），它并未赢得玩家的心，因为它不像《Bejeweled》那样明显更加明朗和欢快（绿颜色或者黄颜色），更适合大众用户短时闲趣娱乐的特征；此外在形状和颜色的布局上也存在着类似的问题。《Blood War》所采用的是形状一致的骷髅形象，而《Bejeweled》则是形象各有特点的宝石形象，在这层上面除了可以利于用



凭借惊人的画面效果，《Indinity Blade 2》发行一个月就获得超过500万美元的收入

户依靠形状和色彩进行辨识外，同样在细节设置上服务于色盲玩家（不能辨识颜色的情况下可以根据形状和特殊符号进行二次识别，同样不影响正常识别能力玩家的游戏体验）。色盲群体通常无法辨别在常人看来非常不同的颜色，而游戏玩法如果涉及与颜色相关的元素时，这个群体的玩家就会遇到麻烦。尽管相对于全体的游戏玩家而言可能没有绝对的基数，却可能导致不良的游戏体验，《Crossfire》的做法是在颜色中添加特殊的记号以作二次区别。

美术和UI布局

在游戏设计中，美术层面的表达可能经常通过两个层面对玩家进行暗示：一个是和玩家本身的文化背景相匹配进行色彩和深浅度的填充，以玩家的熟识度进行情感的快速切入；另一个则是以整体的风格属性去判断和迎合具体群体玩家的认同属性。

具体到风格层面，有些内涵在不同的场合下能即能立竿见影。日本的卡牌社交游戏在本土受到异常的欢迎，但日本风的游戏诸如《Spirit Force》在其他市场就表现得不尽如人意，而《Legend of the Crypitds》、《Zombie Jombie》、《Rage of Bahamut》在摒弃日本漫画风格后同样能在北美获得不俗的用户表现。这种印记的群体属性是相当明显的，包括一些经典IP所授权研发的游戏也都是在整体风格上首先关照了原先的基础用户群，诸如Gameloft推出的《Ice Age Village》、EA推出的《The Simpsons: Tapped Out》、Bandai Namco推出的《One Piece Grand Collection》、

Capcom推出的《The Smurfs Village》、Beeline Interactive推出的《Snoopy's Street Fair》、Mobage推出的《Doraemon Hokkorina》、Beeline Interactive推出的《Shrek's Fairytale Kingdom》等都具有极深的风格属性。

此外的一些创意性作品同样坚持了某种相对独有的核心风格，诸如《Tiny Wings》简易的丘陵纹理以及《Cut The Rope》中典型的Omnom形象。

至于在文化暗示层面，美国动视暴雪的James Portnow认为游戏界面色彩的布局可能影响到用户的进程心态，而色彩心理学之所以能够独立分解出来是因为附着于颜色的符号意义几乎都是文化适应性所产生的结果，用户的普遍看法是：颜色是我们获取图像信息的最基本方式，因此也就必然存在着某些层面的文化参照考量，让玩家从原始的直觉机制切换至某种更正式的方式。例如有颜色意义的背景或符号提示，并最终使颜色和画面具有和玩家对话的沟通效能（颜色选择的艺术性不会因融入暗喻表达而受到削弱并且此表达将适用于各种熟悉的特定情境）。此时似乎又可区分为两个层面：其一是纯色彩的文化内涵，例如以黄颜色作为警示色彩，以灰色显示形象的死亡或者某些特殊的填充物属于未解锁的状态，或者某些高级别的关卡暂时未向用户开放；其二是融入场景因素的情感驱动，主体是和整体的游戏氛围进行比照，诸如场景的色彩映衬，相对阴森的或者相对明朗的方位都能指示用户即将发生的游戏片段的不同的类型属性。

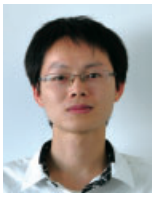
当然这种效能一般也能延续到用户交互界面的层次，诸如鼠标在特定场合的颜色和指向以及弹窗信息中的特别提示都能够在视觉行为中改变玩家的注意力节奏，让其更专注在当前的任务内容而不是被暂时关联性不强的其他内容所分散精力。

画面架构的协调度

Brice Morrison从美学的角度甚至将游戏的音效设计、界面交互设计连同视觉效果设计一起归属在用户审美体验中。众所周知，音效对于整体游戏画面的沉浸表达具有加成作用，特别是在特定的场合诸如阴暗面的场景如果再配合上点惊悚的音乐元素就更加容易将玩家在当刻对画面所表达的蕴涵更淋漓尽致地渗透出来，以致和整体的场景性融合在一起。当然，从格莱美对游戏音乐的认知（诸如Christopher Tin的凭借Sid Meier的《文明IV》中Calling All Dawns的声效获得Best Instrumental Arrangement Accompanying Vocalists ）也正在驱动两者之间的紧密兼容。至于界面交互的元素则如上文提及的以特征化的方式来区别于当前所有游戏的色彩元素，并使之形成相对强势的视觉吸引要素，满足游戏引导玩家进行操作行为的相关意图，或者按照Brice Morrison的说法正是这些交互性界面的存在让游戏的画面元素能够正常流动起来，并具有用户感召力。

但事实上无论是场景、UI、各种游戏填充物所架构的游戏画面是如何表现的，不管美术在单个配件上如何着力，终究当所有东西都混杂奔跑在同一个场景界面时，从协调度的层面就都可以对游戏的当时画面进行解构。

包括场景与每一个不同填充物之间大小比例的协调程度，甚至在特定的情境下可以刻意更改不同配件的比例以显示游戏所想呈现的大状态，例如一个庞然大物所营造的对比差，以产生情境的影响力（比如大块头理所当然的攻击实力，和战斗可能的持久力，或者它有某些特定的收藏价值或者出售价值或者炫耀价值）；包括场景与每一个填充物之间的色泽比照程度，除了画面色彩可能昭示的文化或者情绪牵引力之外，在同一个场景之内当所有填充物混搭时是否还能够保持整体视觉的协调性，当然也有可能是为了突出当刻某个配件的效能而故意做了着色上的偏差；包括游戏画面的整体质感，这可能在协调度（比如整体的画面视觉）之外还需要兼顾更多填充物本身的细节表现力，例如用于界定物品线条的形状、轮廓，不同的曲线形状可能呈现的信息就会产生差异化。



郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

责任编辑：陈博（chenbo@csdn.net）

Cocos2D-X for XNA游戏开发指南（下）

《TweeJump》项目实战

文 / 刘凯威

上一期我们介绍了Cocos2D-X for XNA相对于Cocos2D-X的一些变化，本期以一款名为《TweeJump》的小游戏为例，介绍了Cocos2d开发中的游戏背景的设置、游戏主逻辑、集成OpenXLive SDK等开发实践，并列举了一些Cocos2D中常见的API。

《TweeJump》是一款开源的、采用Cocos2D-iPhone编写的游戏：小鸟以草地为踏板不断地向上跳以获取更多的分数，使用重力感应来控制方向。背景是一块永远不会动的蓝天，中景是缓慢移动的白云，前景是草皮和鸟。通过这款小游戏，我们将学习一些Cocos2D中常见的API，例如怎样加载精灵、设计场景和层、使用动作、菜单按钮等。

游戏背景的设置

在游戏介绍中已得知背景其实是一张不变的蓝色背景图。首先，我们打开Cocos2D for XNA工程，使用Cocos2D-XNA Applocation模板新建一个基于Cocos2D-X for XNA引擎的WP7 XNA项目，命名为TweeJump。

如果你的游戏需要积分榜、游戏成就、社交网络、个人中心、云存储、游戏公告等功能的一个或者多个，可以在新建项目的向导中勾选上Create Cocos2D-XNA project with OpenXLive SDK，在《程序员》6月刊中我们介绍过OpenXLive，后面我们还会介绍相关SDK的使用。

新建完工程，我们可以先运行一下，保证一切正常。顺利的话，编译后可以直接启动WP7模拟器看到Cocos2D-X的LOGO，以及一个关闭按钮。我们把新建工程放到一边，开始填充游戏的背景。

将素材sprites.png复制到TweeJumpContent的Images的目录里，在TweeJump Classes目录下新建Main.cs文件，让Main类继承CCLayer，并重写init()方法。CCLayer的init()方法会在CCLayer实例化时被引擎调用做一些初始化工作：

```
public override bool init()
{
if (!base.init())
{
return false;
}

CCSpriteBatchNode spriteManager =
CCSpriteBatchNode.batchNodeWith-
File("Images/sprites", 10);（注释1）
addChild(spriteManager, -1, (int)tags.
kSpriteManager);

CCSprite background =
CCSprite.
spriteWithTexture(spriteManager.Texture,
new CCRect(0, 0, 320, 480));（注释2）
spriteManager.addChild(background);
Scale(background);
background.position = new CCPoint(240,
400);（注释3）

this.initClouds();

this.schedule(step);
return true;
}
```

我们应该看到sprites.png是由多张图片拼出来的

一张大图，在学习做游戏时经常会看见类似于这样的图，会将很多动作或一堆物件放在同一张图片上，而不是每个都单独一张图。这是为了提高性能：将相关的动作或者出现在同一屏幕上的物体整合在同一张图上，这样就只需载入一次大图就可以了。

上段代码中，标注1把图片sprites变成了CCSpriteBatchNode名字为spriteManager，通过spriteManager.addChild(background)加入背景。

spriteManager主要做什么呢？通常显示一张图片需要把它构建成一个CCSprite然后调用addChild方法将它添加到当前CCScene，CCScene就会调用CCSprite的绘制逻辑帮我们呈现这张图。SpriteBatchNode又是干嘛的呢？当游戏画面中出现了大量的CCSprite时，SpriteBatchNode就能帮助我们提升性能。因为CCSpriteBatchNode中的所有CCSprite的纹理都是同一个，只会被加载一次，所以可以提高游戏的性能。不过这样加入到CCSpriteBatchNode中的CCSprite必须使用同一张纹理图。

上段代码中，标注2和标注3用了CCSprite.spriteWithTexture方法。第一个参数texture精灵通过哪个纹理生成，这里我们选择用spriteManager的Texture。第二个参数是精灵在纹理中的位置和大小，然后调整大小Scale()，因为这部分图的尺寸是320x480的，改变成480x800的填充全屏幕。

最后设置position属性，CCSprite的默认锚点是(.5f,.5f)，所以position属性表示的是精灵中点的位置，这里我们将位置设置到屏幕中央，注意Cocos2D-X是以左下角为坐标原点的。接下来我们绘制白云：

```
private void initClouds()
{
currentCloudTag = (int)tags.
kCloudsStartTag;
while (currentCloudTag
< (int)tags.kCloudsStartTag +
kNumClouds)
{
this.initCloud();
currentCloudTag++;
}

this.resetClouds();
}
```

initClouds方法里循环调用了n次initCloud()来实例化多片云朵。tags.kCloudsStartTag+kNumClouds用来控制initCloud的调用次数，也决定云数量的值。在方法最后的resetClounds()是干什么的呢？会在后文中解释。

```
private void initCloud()
{
CCRect rect = new CCRect();
switch (random.Next() % 3)
{
case 0: rect = new CCRect(336, 16, 256,
108);
break;
case 1: rect = new CCRect(336, 128, 257,
110);
break;
case 2: rect = new CCRect(336, 240, 252,
119);
break;
}
CCSpriteBatchNode spriteManager =
(CCSpriteBatchNode)getChildByTag((int)
tags.kSpriteManager);
CCSprite cloud =
CCSprite.
spriteWithTexture(spriteManager.Texture,
rect);
Scale(cloud);
spriteManager.addChild(cloud, 3,
currentCloudTag);
}
```

从素材sprites.png上可以看出，云有三块，样子各不相同，所以我们绘制纹理的不同区域。

这里想说明一下addChild()方法，它是CCNode的方法。我们知道，CCScene、CCLayer、CCMenu、CCNode、CCSprite CCMenuItem等，都继承自CCNode。Cocos2D的游戏界面最顶层就是一个CCScene，游戏同一时间只有一个CCScene在运行，runningScene代表了当前正在运行的CCScene，通过递归的addChild方法就把界面上一些层级的UI元素转化成程序当中的一棵对象树。

addChild的第二个参数是Z轴深度。如果zOrder=10，则它在zOrder=1上，注意这里的zOrder和XNA中的z-Order是不同的，只是Cocos2D引擎逻辑上的Z轴层级，一般情况下，所有元素的XNA的ZOrder都为0，修改这个值可能会影响Cocos2D中的zOrder。

接下来我们添加一个resetClouds方法，它用于初始化云的位置：

```
public void resetClouds()
{
currentCloudTag = (int)tags.
kCloudsStartTag;
```

```
while (currentCloudTag < (int)tags.
kCloudsStartTag + kNumClouds)
{
this.resetCloud();

CCSpriteBatchNode spriteManager=(CCSpr
iteBatchNode)getChildByTag((int)tags.
kSpriteManager);
CCSprite cloud =(CCSprite)spriteManager.
getChildByTag(currentCloudTag);
CCPoint pos = cloud.position;
pos.y -= 800f;
cloud.position = pos;
currentCloudTag++;
}
}
```

我们通过tag取得initClouds方法创建出来的对象，然后设置pos.y-=800使云处于不可见。再看看resetCloud：

```
public void resetCloud()
{
CCSpriteBatchNode spriteManager=(CCSpr
iteBatchNode)getChildByTag((int)tags.
kSpriteManager);（注释4）
CCSprite cloud =(CCSprite)spriteManager.
getChildByTag(currentCloudTag);（注释5）

float distance = random.Next() % 20 + 5;

float scale = 5f / distance;
cloud.scaleY = scale;
cloud.scaleX = scale;
if (random.Next() % 2 == 1)
cloud.IsFlipX = true;

CCSize size = cloud.contentSize;
float scaled_width = size.width * scale;
float x =random.Next() % (480 + (int)
scaled_width) - scaled_width / 2;
float y =
random.Next() % (800 - (int)scaled_
width) + scaled_width / 2 + 800;

cloud.position = new CCPoint(x, y);
}
```

上段代码中，注释4和注释5取得了当前CCSprite，然后在最后一行代码指定了一个位置信息，中间的那些代码我们暂时先不看。

接下来我们让云朵动起来——在Update里不断地改变每个云朵的位置信息让云动起来。首先需要注册step(dt)这个方法为本类的更新逻辑，这个类除了云动也没其他逻辑了，因此for循环就是不断地改变位置：

```
public void step(float dt)
{
CCSpriteBatchNode spriteManager =
CCSpriteBatchNode)getChildByTag((int)
tags.kSpriteManager);
int t = (int)tags.kCloudsStartTag;
for (int i = t; i < (int)tags.
kCloudsStartTag + kNumClouds; i++)
{
CCSprite cloud = (CCSprite)
spriteManager.getChildByTag(i);
CCPoint pos = cloud.position;
CCSize size = cloud.contentSize;
pos.x += 0.1f * cloud.scaleY;
if (pos.x > 480 + size.width / 2)
pos.x = -size.width / 2;
cloud.position = pos;
}
}
```

这样我们就完成了游戏背景的设置。

游戏主逻辑

下面我们就真正进入游戏逻辑：让小鸟在草地上跳。在完成一切初始化和注册工作后会调用一个方法startGame()，看看startGame()做了什么：

```
void startGame()
{
score = 0;

this.resetClouds();
this.resetPlatforms();
this.resetBird();
this.resetBonus();

gameSuspended = false;
}
```

这个方法首先将所有分数清零，并把所有的CCSprite重置， 游戏结束标识还原默认值。这是为了重用对象，不需要每次进入游戏后重新实例化。在游戏开发中，你应该尽可能多地重用对象。

Cocos2D-X for XNA还没有封装重力感应的。我们需要自己来实现：

步骤1：添加Microsoft.Devices.Sensors引用；

步骤2：创建Accelerometer对象；

步骤3：注册Accelerometer对象的Current-ValueChanged事件；

步骤4：启动Accelerometer对象Start()方法来启动重力感应监听。

过程看起来很复杂，但其代码很简单，在override的init()中添加这几行就行：

```
accelerometer = new Accelerometer();
accelerometer.CurrentValueChanged +=
accelerometer_CurrentValueChanged;
accelerometer.Start();
```

在注册的事件处理方法中可以通过e.Sensor-

Reading.Acceleration.X和e.SensorReading.Acceleration.Y，e.SensorReading.Acceleration.Z来访问当前的加速度的三维偏移值。很多教程里都是注册Accelerometer对象的ReadingChanged事件，但还是建议使用CurrentValueChanged。

TimeBetweenUpdates属性会影响到两次触发CurrentValueChanged事件的时间间隔，因此如果要设置重力感应的感应时间间隔的话，最好还是使用CurrentValueChanged。

```
void accelerometer_
CurrentValueChanged(object sender,
SensorReadingEventArgs<AccelerometerR
eading> e)
{
if (gameSuspended)
{
return;
}
float accel_filter = 0.1f;
bird_vel.x = bird_vel.x * accel_filter +
e.SensorReading.Acceleration.X * (1f -
accel_filter) * 500f;
}
```

游戏逻辑很简单，我们通过重力感应的方法控制小鸟的x坐标的变化，然后小鸟一直在跳，如果触碰到菜地就继续跳，不然就掉下去游戏结束。

```
new void step(float dt)
{
base.step(dt);


if (gameSuspended) return;


CCSpriteBatchNode spriteManager =
(CCSpriteBatchNode)getChildByTag((int)
tags.kSpriteManager);
CCSprite bird = (CCSprite)spriteManager.
getChildByTag((int)tags.kBird);


bird_pos.x += bird_vel.x * dt;


if (bird_vel.x < -30f &&
birdLookingRight)
{
birdLookingRight = true;
bird.IsFlipX = true;
}
else if (bird_vel.x > 30f &&
birdLookingRight)
{
birdLookingRight = true;
bird.IsFlipX = false;
}
```

Step(float dt)的代码很长，上面是一部分，首先调用了父类的step(dt)，因为我们还需要云是移动的，然后找到这只鸟的对象，再判断重力感应的左右偏向让小鸟转头：

```
CCSize bird_size = new CCSize(bird.
contentSize.width * sx, bird.
contentSize.height * sy);
float max_x = 480 - bird_size.width / 2;
float min_x = 0 + bird_size.width / 2;
if (bird_pos.x > max_x) bird_pos.x =
max_x;


if (bird_pos.x < min_x) bird_pos.x =
min_x;


bird_vel.y += bird_acc.y * dt;（注释6）
bird_pos.y += bird_vel.y * dt;（注释7）


CCSprite bonus = (CCSprite)
spriteManager.getChildByTag((int)tags.
kBonusStartTag + currentBonusType);
if (bonus.visible)…
```

计算小鸟的体积，注释6和注释7不断地在改变小鸟y的位置，其实这里才是真正的跳的逻辑：

```
int t;
if (bird_vel.y < 0)
{
t = (int)tags.kPlatformsStartTag;
for (int i = t; i < (int)tags.
kPlatformsStartTag + kNumPlatforms; i++)
{
CCSprite platform = (CCSprite)
spriteManager.getChildByTag(i);


CCSize platform_size = new
CCSize(platform.contentSize.width * sx,
platform.contentSize.height * sy);
CCPoint platform_pos = platform.
position;


max_x = platform_pos.x - platform_size.
width / 2 - 10;
min_x = platform_pos.x + platform_size.
width / 2 + 10;
float min_y = platform_pos.y + (platform_
size.height + bird_size.height) / 2 -
kPlatformTopPadding;


if (bird_pos.x > max_x &&
bird_pos.x < min_x &&
bird_pos.y > platform_pos.y &&
bird_pos.y < min_y)
{
this.jump();
}
}
```

如果在落下后和草地有碰撞则执行jump方法：

```
void jump()
{
bird_vel.y = 583f + Math.Abs(bird_vel.
x);
//bird_vel.y = 53f + Math.Abs(bird_vel.
x);
}
```

如果小鸟y的位置超出了屏幕的一半，则开始for循环：resetCloud(),resetPlatform(),resetBonus()。

集成OpenXLive SDK

如果你的游戏需要社交功能，可以选择使用OpenXLive SDK。前面如果勾选了OpenXLive SDK，现在我们的项目添加好了引用就可以很方便地使用它。

我们看到Cocos2D-X for XNA就是一个GameComponent。而OpenXLive SDK也是一个GameComponent，打开主游戏类Game1（源代码中叫GameMain），就是找到继承于Microsoft.XNA.Framework.Game的类。

对比我们的《TweeJump》项目和示例程序HelloCocos2D的Game1.cs文件，多的几句代码就是集成OpenXLive的方式。

Game1.cs中字段：

```
//#error Please full your game Secret
Key in below code
private string APISecretKey =
"Y8eUF6f5xXEGSuw3DkKntbhQ";
public static XLiveFormManager manager;

// Create XLive FormManager
manager = new XLiveFormManager(this,
APISecretKey);
manager.OpenSession();
// Add XLive FormManager in Components
Components.Add(manager);
```

这里需要注意的是OpenXLive的GameComponents的Add必须放在Cocos2D for XNA的游戏引擎的GameComponents之前。

在LoadContent()里加载资源给OpenXLive SDK UI添加背景和退出UI的事件。

```
protected override void LoadContent()
{
Texture2D background = this.Content.
Load<Texture2D>(@"images\OpenXLive");
manager.Background = background;
manager.UIExitingEvent += new
EventHandler(manager_UIExitingEvent);
base.LoadContent();
}

void manager_UIExitingEvent(object
sender, EventArgs e)
{
CCDirector.sharedDirector().
runningScene.visible = true;
CCDirector.sharedDirector().resume();
}
```

勾选后上面是在新建项目的时候自动添加到Game1.cs中的代码。

下面以MainMenu为例教你在游戏中使用OpenXLive SDK。

```
CCMenuItemImage Leaderboards =
CCMenuItemImage.itemFromNormalImage(
@"images\Leaderboards",
@"images\Leaderboards1",
this,
new SEL_MenuHandler(LeaderboardsCallba
ck));
Leaderboards.position = new CCPoint(0,
20);


CCMenu OpenXLiveMenu = CCMenu.
menuWithItems(Lobby, Leaderboards,
Achievements);
OpenXLiveMenu.alignItemsHorizontallyWith
Padding(40);
OpenXLiveMenu.position = new
CCPoint(240, 60);
this.addChild(OpenXLiveMenu, 1);
```

OpenXLive.Forms.XLiveFormFactory.Factory.ShowForm("Lobby")呈现我们需要的窗体这里是Lobby，除了“Lobby”还有“Startup”、“Leaderboard”、“Achievements”、“Logon”、“MyCenter”、“FirendsList”、“OnlinePlayer”、“Pause”、“OpenXLiveGames”。

TweeJump源代码中使用了“Achievements”、“Leaderboard”、“Lobby”这三个常用功能。在Show出窗体后务必把“游戏暂停当前Scene”隐藏。在回来时也务必把“游戏开始当前Scene”显示。

总结

游戏逻辑比较简单，没有用到Cocos2D的太多功能，例如地图、字体、动作、场景切换、粒子系统等。但我们仍然能够感受到，在使用Cocos2D时，我们将精力放在了游戏逻辑中，通常只需要几句代码就可以使用某个功能。而我们完全不用去考虑XNA绘制、矩阵变换、摄像机等这些游戏开发底层的技术，因此大大降低了开发Windows Phone游戏的难度。



刘凯威

Cocos2D-X for XNA开发者，关注移动互联网，喜欢游戏。

责任编辑：陈博（chenbo@csdn.net）

iOS即时语音聊天技术实践

文 / 张天虹

本文讲解了如何在iOS设备上实现语音聊天应用开发，涵盖了语音应用开发中的语音录制、播放、编解码等技术。

语音聊天的基本原理

使用语音聊天应用时需要用户做的有录音、发送、播放语音这三个步骤，那么对应的从实现技术的角度我们就需要关注如何在iOS中进行语音录制、语音数据传输以及语音播放。

由于移动网络流量比较珍贵，所以我们在语音数据传输前就必须考虑语音编码的环节，同时在语音数据接受之后进行语音解码。因此，我们需要考虑的技术包括语音录制、语音播放、语音传输以及语音编解码。

语音录制

iOS SDK提供的录音API是非常友好的，因此实现录音功能是非常简单的工作。但iOS的录音API并不是万能的，我们需要首先了解一下iOS支持的录音格式，然后详细介绍录音API的使用，以及实现录音时音量变化的UI效果，最后我们还会提及更高级的面向音频流的录音方式。

iOS默认支持的语音录制格式

iOS SDK 5.0目前支持的语音录制格式包括：AAC、ALAC、iLBC、IMA4、Linear PCM、μ-law和a-law。每种音频格式都有不同的特点，主要的区别在于音质的品质和压缩体积的变化。在不同的使用场景中，对音质和压缩率的要求都不尽相同，自然也衍生出了许多音频格式标准，iOS也为我们提供不同的音频录制格式来满足不同开发者的需求。

简单的录音API——AVAudioRecorder

iOS SDK的AVFoundation.framework里提供了AVAudioRecorder这个用于录音的类。这个类提供了面向目标文件的录音方式，录制出来的音频数据最终会生成一个录音文件被保存。录音主要分为四个关键步骤。

建立录音的目标文件。首先建立文件路径，然后生成一个NSURL用于指定录音的目标文件。

```
NSURL *audioRecorderUrl=[[NSURL alloc] initFileURL
WithPath:RECORDER_TEMP_FILE_NAME];
```

设置录音文件的信息。在录音之前需要确定录制的音频数据的基本信息，例如录音文件的格式、采样率、通道数、线性采样位数等。这里的采样率和线性采样位数决定了录音采样的语音品质，采样率和采样位数越高，录制出来的音频数据的精细程度就越高。

```
NSMutableDictionary *recoderSetting=
[[[NSMutableDictionary alloc] init] autorelease];
[recoderSetting setValue:[NSNumber numberWithInt:
kAudioFormatLinearPCM] forKey:AVFormatIDKey];
[recoderSetting setValue:[NSNumber numberWithFloat
:8000] forKey:AVSampleRateKey];
[recoderSetting setValue:[NSNumber numberWithInt:1
] forKey:AVNumberOfChannelsKey];
[recoderSetting setValue:[NSNumber numberWithInt:
16] forKey:AVLinearPCMIsFloatKey];
```

实例化AVAudioRecorder。指定好目标文件

路径和录音格式信息之后就可以实例化AVAudioRecorder对象了，同时可以捕捉录制过程中产生的错误。

```
AVAudioRecorder *recorder=[[AVAudioRecorder alloc]
initWithURL:audioRecorderUrl settings:recoderSetti
ng error:&error];
```

准备并开始录音。先是准备录音，如果调用了prepareToRecord方法，麦克风会处于待命状态并建立好与目标文件的链接，然后调用record方法开始录音。当然如果不调用prepareToRecord也可以直接录音，但这样会导致每次录音时都需要等一会儿，因为这时是在录音时才建立与目标文件的连接。

```
[recorder prepareToRecord];
[recorder record];
```

录音时捕捉话筒音量

在设计录音的UI时，我们往往会使用一种音量波动的效果，这样可以给用户更友好的操作反馈。iOS提供了音量检测的功能，我们可以通过音量检测出录音或者播放时的音量变化，从而为用户展现一个动态的录音动画。

```
[NSTimer scheduledTimerWithTimeInterval:0.05 target
:self selector:@selector(levelTimeCallback:) userIn
fo:nil repeats:YES];
```

首先，我们开启音量检测功能recorder.meteringEnabled=YES，然后定义一个计时器，不断地获取当前的音量数据。

接下来通过定时器的CallBack刷新当前的音量数据，可以获取到这段时间某一个通道的平均音量数据和平均峰值音量数据。最后将这些数据体现在UI上，形成一个动态变化的效果。

```
- (void)levelTimeCallback:(NSTimer *)timer
{
    [recoder updateMeters];
    CGFloat averagePower=
                [recoder averagePowerForChannel:0];
    CGFloat peakPower=
                [recoder peakPowerForChannel:0];
    //更改UI的图片效果
}
```

面向音频流的录音方式

除了面向文件的录制方式，iOS还提供了面向音频流的录制方式，面向音频流的录制方式可以让开发人员更好地控制音频的录制过程，比如在录制数据和保存数据的过程中对数据进行分析或转换等操作。

面向音频流的录音方式相对比较复杂，这里会用到Audio Queue Services、Audio File Services、Audio Session Services等服务。具体的实现细节可以参考iOS官方Demo“SpeakHere Demo”【注：链接为https://developer.apple.com/library/ios/#samplecode/SpeakHere/Introduction/Intro.html】。

语音编解码

实际上，iOS系统音频默认录制的数据可以是PCM（Puise-code modulation）的格式，这种格式的音频数据未经压缩，所以音质效果非常好，同时体积也较大。因此我们在录制完音频后直接将PCM数据通过网络传输是非常消耗网络资源的，而且效率很低，所以我们需要对音频进行编码。

iOS音频数据录制和编码过程

图1所示的是iOS录制AAC音频数据文件的过程，大致分为三个步骤：首先我们开发的应用调用iOS设备的麦克录制PCM音频数据；然后PCM数据通过Codec（coder-decoder）编码解码器将PCM数据编码为AAC数据；最后将AAC数据保存到硬盘文件中。图2所示的是一个相反的过程，其中解码过程也是通过Codec进行。

Codec

如图1所示，音频的编码和解码依赖的都是Codec，iOS之所以支持有限的音频录制格式，正是因为iOS只提供有限的Codec，它们是：Apple LossLess、iLBC、IMA/ADPCM、Linear PCM、

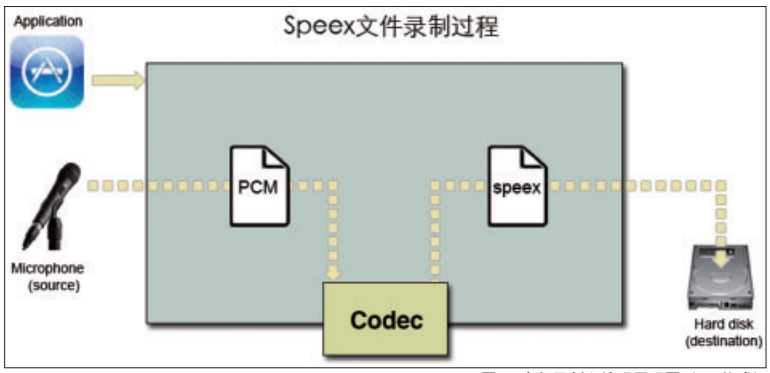


图1　音频录制和编码原理图（AAC格式）

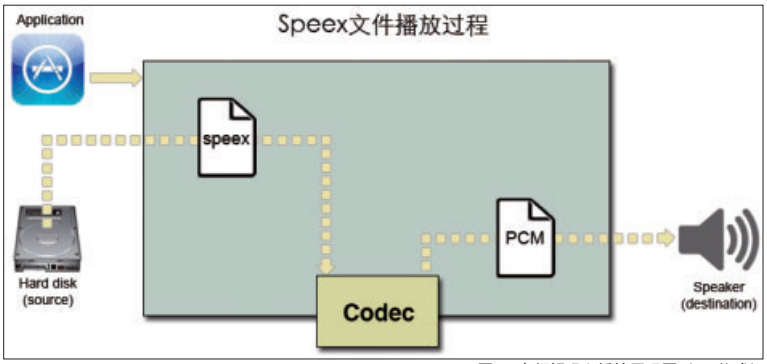


图2 音频解码和播放原理图（AAC格式）

μ-law和a-law。它们所编码的格式正好对应了支持的录制格式。但目前常用的音频格式种类繁多，且不同音频格式适合的场合也不尽相同，那么实际使用中我们想录制其他格式的音频该怎么办呢？例如比较常用的MP3、WMA、MIDI、Ogg、Speex等。

引入开源的Codec

实际上，目前有非常多的开源音频Codec可供我们使用，例如针对MP3的Codec有lame，无损压缩的有flac，适合语音聊天的speex等。

那如何在我们的项目中使用这些开源的Codec呢？首先了解一下iOS的开发环境XCode。XCode里面支持模拟器编译调试以及真机编译调试，因此XCode编译时需要支持的处理器架构包括i386（iOS Simulator）、ARMv6（iPhone 2G/3G、iPod 1G/2G）、ARMv7（ARMv6以后机型）。

开源的Codec大多提供C语言的开源项目下载，并提供了跨平台的makefile工程，所以我们需要下载并编译适合XCode的动态链接库。下面以编译speex开源库为例进行介绍。

编译适合i386的链接库

```
RainbowdeMacBook-Pro:~ Rainbow$ /configure -prefix
=/Users/Rainbow/speex/i386 -host=i386-apple-darwin
 -build=x86_64-apple-darwin10.8.0 CC="/Developer/P
latforms/iPhoneSimulator.platform/Developer/usr/bi
n/gcc -std=c99 -arch i386 -isysroot /Developer/Pla
tforms/iPhoneSimulator.platform/Developer/SDKs/iPh
oneSimulator4.3.sdk/"
```

编译适合armv6的链接库

```
RainbowdeMacBook-Pro:~ Rainbow$ /configure -prefix
=/Users/Rainbow/speex/armv6 -host=armv6-apple-darw
in -build=x86_64-apple-darwin10.8.0 CC="/Developer
/Platforms/iPhoneOS.platform/Developer/usr/bin/gcc
 -std=c99 -arch armv6 -isysroot /Developer/Platfor
ms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.3.sd
k/"
```

编译适合armv7的链接库

```
RainbowdeMacBook-Pro:~ Rainbow$ /configure -prefix
=/Users/Rainbow/speex/armv7 -host=armv7-apple-darw
in -build=x86_64-apple-darwin10.8.0 CC="/Developer
/Platforms/iPhoneOS.platform/Developer/usr/bin/gcc
 -std=c99 -arch armv7 -isysroot /Developer/Platfor
ms/iPhoneOS.platform/Developer/SDKs/iPhoneOS4.3.sd
k/"
```

合并链接库供XCode使用

```
RainbowdeMacBook-Pro:~ Rainbow$  lipo -create i386
/lib/libspeex.a armv6/lib/libspeex.a armv7/lib/lib
speex.a -output libspeex.a
```

最后将合并的动态链接库libspeex.a导入XCode，并在User Header Search Paths里面加入speex的头文件。这样在编写代码时才能导入speex.h调用speex API。

使用Codec编解码音频数据

首先来看Codec编码的过程，因为需要使用自定义的Codec进行编码，所以录制时需要将默认的录制的音频格式设为PCM，设置AVFormatIDKey=kAudioFormatLinearPCM。根据录制方式的不同在解析时需要分为两种情况。

第一种是以文件的方式录制，这种情况录制出来的音频文件默认是WAVE格式，那么在使用Codec解码前需要将文件的头信息和数据块信息去掉，将最终取到的音频数据进行编码。例如图3中需要取出Digital Audio Samples的数据。

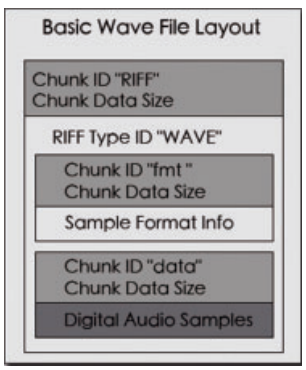


图3 WAVE文件结构

第二种是以音频流录制的方式，这种方式录制的音频数据没有包含别的信息数据，可以直接进行编码。

编码时根据不同的Codec的特点，需要设置一些特定的编码属性，然后调用Codec的encode方法即可实现编码的过程。

<Encoded Data>=codec_encode(<Digital Audio Samples Data>)

这里需要注意的是编码时是按帧进行的，还可以定义每一帧的长度，然后遍历PCM格式的音频数据进行编码。编码完成之后就得到了相应Codec的raw数据。编码后的raw数据可通过网络传输到另外一台设备进行解码操作，但如果希望编码后的音频数据在别的平台播放，就需要为raw数据添加文件格式信息。文件格式信息根据不同的Codec有不同的结构。

音频数据经过服务器中转后发送到另一台iOS设备上，接下来就需要对接收到的音频数据进行解码了。由于之前提到编码后的数据有可能含有文件结构信息，所以解码的第一步就是去掉文件接口信息，最后得到编码后的音频raw数据。如果没有文件结构信息则可以直接进行解码。

解码的过程正好和编码相反，在设定Codec的一些解码属性之后调用Codec的decode方法实现解码，原理同编码都是需要按帧进行。

<Digital Audio Samples Data>=codec_decode(<Encoded Data>)

解码后的数据恢复为了PCM音频数据，到这里我们就完成了在iOS中完成编码和解码的过程。

语音播放

相对于语音的录制，实现语音播放更简单。iOS只支持部分音频格式播放，其中包括：AAC、ALAC、HE-AAC、iLBC、IMA4、Linear PCM、MP3、μ-Law和α-Law等，这些音频数据都需要以文件的格式进行播放，这样才能让播放器知道文件的基本信息。

例如上文解码出的PCM数据需要重新组装为WAVE文件后才能让AVAudioPlayer进行播放。iOS提供的AVAudioPlayer使用起来也非常简单，只需传入需要播放的音频数据，然后播放即可。

```
AVAudioPlayer *player=[[AVAudioPlayer alloc]
                       initWithData:soundData
                       error:&error];
[player prepareToPlay];
[player play];
```

Audio Session

iOS提供一个管理音频上下文的单例对象AVAudioSession，它可以帮助我们轻松管理音频的输入和输出。例如在iPod正在播放音乐时启动应用，当这个应用也要使用音频时，iPod的音乐是继续播放还是停止播放呢？还有就是应用的音频在锁屏时是否继续播放？这些问题都可以由AVAudioSession去管理。

Audio Session主要处理三种音频情况：静音或锁屏的情况下是否停止播放音频、是否支持其他应用同时使用音频、是否支持音频的输入和输出。针对这三种情况，iOS提供了六种可以选择的Category，默认为AVAudioSessionCategorySoloAmbient。如果需要设置为其他的Category，只需进行要简单的调用。

```
[[AVAudioSession sharedInstance] setCategory:AVAudio
SessionCategoryAmbient error:&error];
```

语音的编解码过程是语音处理的一个难点，因此，了解Codec的编解码原理有助于在语音编解码的过程中进行更多的优化工作，从而提高整个应用的效率。



张天虹

北京爱图腾科技公司高级iOS开发工程师，具有丰富的阅读、工具以及聊天类应用开发经验。

责任编辑：陈博（chenbo@csdn.net）

中国首台云电脑全面解析

天霆云计算董事长谈天霆专访

记者 / 林溪

谈天霆，天霆云计算科技（上海）有限公司董事长。他14岁开始创业，16岁开始接触虚拟化技术，2006年创建天霆云计算科技有限公司。近期，《程序员》杂志对谈天霆进行了专访，请他全面解析中国首台云电脑。

云电脑—全新的IT服务模式

《程序员》：天霆是云电脑概念的首创者，您能具体介绍一下这个全新的IT服务吗？

谈天霆：云电脑服务包括天霆云终端、CHP技术和云端资源，融合了VDI（Virtual Desktop Infrastructure，虚拟桌面基础架构）、DaaS（Desktop as a Service，桌面即服务）和云终端三者的优势。VDI是在2009年流行起来的一种解决方案，DaaS是一种服务，云终端则是能够接收云服务的硬件产品。用户通过在我们网站上订购一个资源池即可配置自己的桌面，按需购买，按需使用，分配灵活，实现了从购买硬件到购买资源的转变。这不仅方便，而且具有很强的灵活性和可伸展性。

《程序员》：云电脑的应用场景是怎样的？

谈天霆：云电脑的应用场景非常广泛。我们有一个通用的解决方案，它可以代替PC。目前，我们的用户主要有政府机关、企业、学校、呼叫中心和制造业等。在未来两三年，我们还会逐步将云电脑推向个人市场，慢慢将云电脑普及化。

《程序员》：云电脑能够为个人和企业带来哪些改变？

谈天霆：对个人用户而言，云电脑的费用更低、使用更便捷、更省电、更安全、体积更小、升级更方便，终端设备的寿命更长。以往要升级PC机的硬件，需要先购买硬件，然后进行组装和软件安装；而使用云电脑，只需在网站上购买资源进行扩展即可，非常方便。同时，我们提供365天、7×24小时的电话服务及在线支持服务，可及时与用户进行沟通，帮助用户及时解决问题，而无须等厂商上门提供服务。

对企业用户来说，PC机不仅可用性低，而且业务连续性和安全性不高，运维也费时费力。在某台机器宕机时，很难将机器迁移到其他物理主机上，来实现平稳的过渡和不间断运行。云电脑可以提供高可用性来保障企业的业务连续性。当用户的某台服务器宕机时，可以在3秒钟内将用户桌面迁移到其他服务器，保障正常运行，用户完全感觉不到，这样可以大大提高业务连续性。同时用户访问的所有资源都在数据中心进行集中管理，统一对用户进行交付，对终端进行交付。这种模式大大提高了数据安全性，降低了企业的维护成本，增强了可管理性。

与传统的VDI解决方案相比，云电脑的成本降低了2/3左右，不需要专人维护，且能够提供高清的交付及流畅的用户体验，是非常不错的选择。

核心技术解析

《程序员》：除了刚才提到的客户端软硬件，天霆是否也同时提供云端的计算和存储服务？

谈天霆：实际上，云电脑本身就包含了存储和计算，因为桌面服务系统需要用到计算、内存、网络和存储资源。在存储方面，我们做到了统一的数据存储，这在整个体系中非常重要。传统的PC机中通常有C盘、D盘等，而云电脑的存储模式与此不同，它主要提供一个类似于云存储的服务，集中保存用户的所有数据，用户可以随时随地通过任何设备输入用户名提取数据。

《程序员》：CHP技术是完成数据中心与云电脑之间交互的关键点，这在技术上是如何实现的？

谈天霆：CHP技术是天霆完全自主研发的一项远程高清呈现技术，能为用户提供一套完整的高用户体验功能，具有高安全性、高清体验、高速外设连接、低带宽连接、广域网接入和2D/3D支持等特性。CHP技术可以在性能和低带宽之间达成完美平衡。CHP技术可提供网络和性能优化功能，通过任何网络交付最出色的用户体验，包括低带宽和高延时WAN连接。其中有一项基于带宽动态调整传输质量的技术，它主要基于广域网的环境状况动态调整图像传输质量，以保证用户程序的正常运行。我们的即插即用技术能支持所有USB外设和COM端口等，多媒体高清交付技术能确保AutoCAD、ProE、Photoshop等软件在云电脑上正常流畅运行，获得近似于在PC机上的使用体验。



在第四届中国云计算大会上，CSDN专访了谈天霆（图左），共同探讨了中国首台云电脑的技术实现和发展前景

《程序员》：在系统的安全性和稳定性等方面有哪些难题，又是如何解决的？

谈天霆：就稳定性而言，我们在生产环境下进行了严格的稳定性测试，通过各种手段确认硬件的稳定性能达到7×24小时无间断正常运行。同时，从实际部署实施的多个试点单位的反馈来看，我们的产品也确实做到了7×24小时无间断运行。在安全性方面，因为云电脑服务的模式本来就是将数据集中管理，所以安全性自然也就提高了。另外我们在底层做了病毒防护，并将多种安全策略加载在底层上来确保安全性。

市场前景展望

《程序员》：大家都知道亚马逊的S3存储是按需付费的，天霆的云存储是以怎样的方式提供给用户的，费用如何？

谈天霆：天霆的云存储跟亚马逊的S3云存储的应用场景有较大区别。我们的云存储主要用来代替原来PC机的硬盘，让用户在任何地方通过不同的设备都可以看到自己的数据，最主要是在桌面上应用这个存储系统。而亚马逊的云存储则主要针对服务器。在费用方面，云电脑产品的费用中包含了云存储的费用，不同型号的云电脑有不同的费用。用户可以根据自己的需求灵活地购买资源，按需付费，真正做到弹性计算的业务模式，IT资源的使用将像用电、用水一样精细可控。

《程序员》：天霆云电脑的推广情况怎么样？

谈天霆：今年差不多有十万台的出货量，主要用在呼叫中心行业、金融行业、制造行业及企业办公行业。

云计算在桌面解决方案上最大的困难是带宽的问题，因为云计算对网络依赖度较大。目前，按照我们的技术来说，要实现桌面系统的正常办公，需要的带宽是20KB/s左右；播放视频或在线维护需要的带宽是2MB/s左右，普通广域网带宽就可以满足这个需求。我们的愿景是为每个人提供IT服务，让IT服务每个人。

目前，云电脑市场还处于培育期。从今年开始两三年之内应该是私有云桌面呈现快速增长的时期。两三年之后，随着用户的认知度提高和市场的成熟及整个产业的发展，云电脑服务会真正呈现出爆发式增长的趋势。

记者后记：这次采访，让我有机会近距离地接触谈天霆这位年仅22岁的董事长。在他身上我感受到了特属于年轻人的那份坚定和执著，然而更让我印象深刻的是他身上的那份原不属于这个年龄的成熟和沉稳。相信在他的带领下，天霆云电脑将拥有卓越的明天。

公共IaaS服务的实测与选择

文 / 蒋清野

本文基于多个公共IaaS服务的实际测试数据，从架构、存储等多个方面，阐释了影响IaaS服务性价比的因素，同时给出了中肯的建议。

本文是我这几年来学习和实践的总结，内容比较粗浅，抛砖引玉吧。

资源超售

通常，虚拟化的计算资源和实际的计算资源是不对等的。目前，主流的虚拟化技术都支持超售（over commit）功能。所谓超售，就是供用户使用的虚拟资源数量大于实际的资源数量。

以CPU为例，在一台物理机上，操作系统所看到的CPU个数=物理CPU个数×每个CPU的核心数量×每个核心的线程数量。假设我的笔记本配置一颗双核CPU，每个CPU核跑两个线程，那么操作系统看到的就是4颗CPU。在没有超售的情况下，这样一台物理机最多只能够支持4台虚拟机。在超售的情况下，这样一台物理机所能够提供的虚拟机数量取决于所使用的虚拟化技术。KVM可以将一个物理CPU核虚拟化成10颗虚拟CPU，这样配置一颗双核CPU的物理机最多可以支持到20台虚拟机。

不仅CPU可以超售，内存也是可以超售的。仍然以KVM为例，一台KVM虚拟机相当于宿主机上的一个进程，它所占用的物理内存大小是动态变化的。当虚拟机需要使用内存时，宿主机才将所需的内存分配给虚拟机进程。不管创建虚拟机时给虚拟机所指定的内存大小是多少，宿主机上的虚拟机进程所占用的内存只比虚拟机上的操作系统所使用的内存多一点点。

当一个虚拟机进程长期处于非活跃状态时，它所占用的内存还会被转移到宿主机的交换分区（SWAP）上。当所有虚拟机请求的内存总数超过宿主机所配置的内存大小时，宿主机也会用交换分区的空间来充数。可以想象，在内存超售的前提下，如果所有的虚拟机同时运行内存密集型应用的话，就会导致较大的性能问题。

值得庆幸的是，虚拟机的CPU和内存使用率通常不会达到100%，这就给IaaS服务提供商带来了很大的机会。一方面，将大量负载较轻的虚拟机集中到少量物理机上，能够大幅提高资源利用率；另一方面，空闲出来的物理机可以进入休眠状态，从而达到节能的目的。

一般来说，客户的虚拟机对CPU和内存的需求会随着时间发生变化，因此IaaS服务提供商需要有一套监控分析、在线迁移和负载调整的工具才能够在实现超售的同时提供较好的用户体验。可以这么认为，在IaaS服务的时代，超售能力将会是IaaS服务提供商的核心竞争力之一。

对于IaaS的最终用户来说，服务提供商所采取的

超售措施是不可见的。图1中的右侧内容列出了三种可能的虚拟机配置参数，其中CPU参数指的是虚拟机可以使用的CPU核的数量，VCPU参数指的是虚拟机所看到的CPU数量，MEMORY参数指的是虚拟机所看到的内存数量。在没有超售的情况下，虚拟机可以使用1个CPU核，提供1颗虚拟的CPU；在1：2超售的情况下，虚拟机只能够使用0.5个CPU核，提供1颗虚拟的CPU；在1：10超售这种极端情况下，虚拟机只能够使用0.1个CPU核，提供1颗虚拟的CPU。即使不考虑宿主机实际的物理内存是否够用的问题，这三种虚拟机的性能也肯定是不同的。但从终端用户的角度来看，这三种虚拟机的配置是相同的。



（1）没有超售，4 台虚拟机

CPU = 1.0
VCPU = 1
MEMORY = 512
OS = [ARCH = x86_64]

（2）1:2 超售，8 台虚拟机

CPU = 0.5
VCPU = 1
MEMORY = 512
OS = [ARCH = x86_64]

（3）1:10 超售，40 台虚拟机

CPU = 0.1
VCPU = 1
MEMORY = 512
OS = [ARCH = x86_64]

图1　资源超售

以前，我们买了台物理机搬回办公室，里面到底有几根内存条、几颗CPU，拆开机箱一看就知道了。现在我们通过IaaS服务买了台云主机，网页上写着2颗CPU外加2GB内存，但我们对网页后面到底是什么一无所知。可以说，**物理机是个白盒子，云主机是个黑盒子。云主机的配置已经退化成一个符号，不再是一个具有确定意义的参数。**不但不同IaaS服务提供商之间的产品无法基于配置进行比较，同一IaaS服务提供商的不同产品之间也很难基于配置来进行比较。这就是基于公共IaaS服务开发应用所面临的挑战。

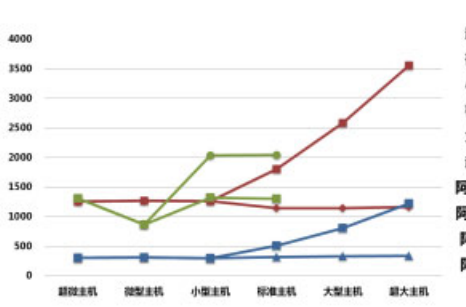
基准测试

那么，我们怎样知道某个IaaS服务提供商的云主机是不是能够满足我们的要求呢？要回答这个问题，只需进行有针对性的基准测试。图2中对比了三组云主机的UnixBench基准测试结果，其中A组（红色曲线）是我自己实验室中的云主机，B组（蓝色曲线）是盛大云正式提供的云主机，C组（绿色曲线）是阿里云正式提供的云主机（阿里云的主机配置和其他两组都不一样，勉强按照比较相近的配置归类）。每台主机的测试结果包括两个数据：一个是单线程测试结果，另一个是多线程测试结果（云主机上有几颗虚拟CPU，就有几个并发的测试线程）。

从图1中的左侧内容可以看出：不管云主机上有几颗虚拟CPU，单线程测试结果是基本一致的，它反映的是单颗虚拟CPU的处理能力；内存大小对测试结果基本上没有影响；当虚拟CPU的数目增长1倍时，测试结果增长0.5倍；盛大云主机的测试数据比我实验室中的云主机的测试数据要稍逊一筹。

UnixBench测试结果反映的是一台主机的综合性能，它与被测试主机的CPU、内存、存储和操作系统有直接关系。我的宿主机配置的是Intel的Xeon E5620处理器，盛大云的宿主机配置的是AMD的Opteron 6172处理器。从cpubenchmark.net提供的数据看，Opteron 6172的整体处理能力比Xeon 5620高50%以上，单核处理能力也比Xeon E5620高一点。

在内存方面，被测试云主机的内存配置是一致的。在操作系统方面，我的云主机和盛大云主机都运行64位Ubuntu 10.04服务器版（阿里云主机运行64位Ubuntu 10.10服务器版）。较大的差异可能在存储方面，我的存储基本上处于空闲状态，盛大云的存储可能负载比较高，但不应该对测试结果产生如此大的影响。比较合理的推测是



	CPU	内存	月价格
超微	1	0.5 GB	34.75
微型	1	1 GB	69.5
小型	1	2 GB	139
标准	2	4 GB	278
大型	4	8 GB	556
超大	8	16 GB	1112
阿里EA	1	0.5 GB	99
阿里EB	1	1.5 GB	199
阿里A	2	1.5 GB	399
阿里B	2	2.5 GB	599

* 阿里云包含带宽价格

实验云（A 系列）
Xeon E5620 @ 2.40 GHz
4 核心 x 2 线程
性能：4693（586 / 线程）

盛大云（B 系列）
Opteron 6172 @ 2.10 GHz
12 核心
性能：7784（648 / 核心）

阿里云（C 系列）
Xeon E5645 @ 2.40 GHz
6 核心 x 2 线程
性能：6906（576 / 线程）

图2　基准测试——UnixBench

盛大云采取了某种超售策略，限制了虚拟CPU的处理能力和虚拟机实际上能够使用的内存数量，从而导致测试结果大幅降低。

阿里云的CPU和我所使用的CPU单核性能是类似的，UnixBench测试得分也相当接近。值得注意的是，经济B型主机的性能与经济A型主机相比有明显的下降（图2中绿色曲线的下降段），而经济B型主机和经济A型主机的配置差别在于经济B型比经济A型多了1GB的内存。也就是说，经济B型主机的性能应该比经济A型主机更好才对，但实际的测试结果正好相反。

为了解开这个谜团，我查看了一下经济A型主机和经济B型主机的CPU类型，发现经济A型主机用的是Xeon E5645，是一款比较好的CPU；经济B型主机用的是Xeon E5620，是一款比较差的CPU。但新的问题又来了，阿里的经济B型主机和我实验室中的微型主机采用的是同样的CPU，内存还比我的微型主机多了0.5GB，为什么在性能上还比我的微型主机差许多呢？从图2中绿色曲线下降段的点来看，我想，比较合理的推测还是阿里的经济B型主机采用了某种超售策略，从而限制了主机的整体处理能力。

表面上配置较高（并且价格也更高）的云主机，由于采用了相对低端的CPU，其性能比表面上配置较低的云主机更低。这就验证了前面所讲述的：云主机的配置已经退化成一个符号，不再是一个具有确定意义的参数。不但不同IaaS服务提供商之间的产品无法基于配置进行比较，同一IaaS服务提供商的不同产品之间也很难基于配置来进行比较。通过这个例子我们还可以看出，阿里云在基础构架服务产品的设计方面相对较弱。

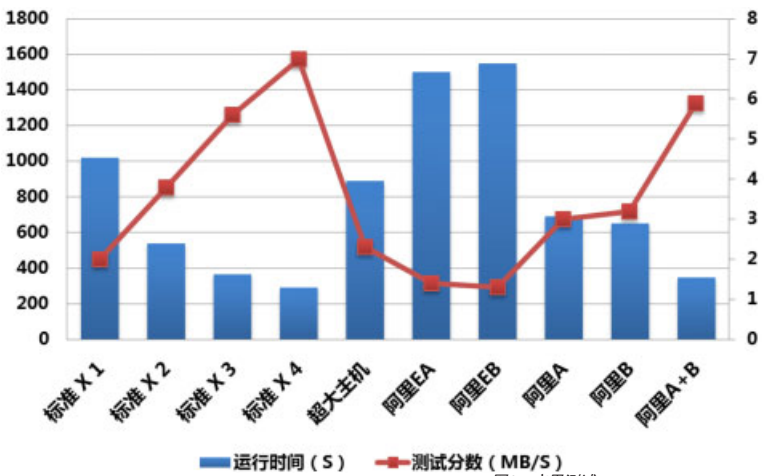


图3 应用测试——Hadoop wordcount

应用测试

UnixBench测试结果在一定程度上反映了一台主机的整体性能。但UnixBench毕竟只是一套简单测试的组合，我们所开发或者使用的应用要比UnixBench复杂得多。这里就以目前比较流行的Hadoop为例，看看它在不同的云主机上的性能如何。在我们下载到的Hadoop软件包中包含了一些作为例子的应用，其中的wordcount应用估计很多人都运行过。这个应用读取输入目录中的文件，计算这些文件中一共有多少个不同的单词以及它们出现的次数。在这个测试中，我使用了三个700MB左右的文件作为测试数据，测试数据一共有2GB左右。我们在盛大云和阿里云的不同云主机上运行wordcount应用对这些测试数据进行分析，并分别记录完成分析所需要的时间。然后用测试数据的大小除以运行时间，代表单位时间内所处理的数据大小（MB/s），用来代表被测配置的测试分数。

我们一共测试了几种不同的配置，分别为由一到四台盛大标准主机组成的Hadoop（伪）集群，一台盛大超大主机组成的Hadoop伪集群，由一台阿里EA、阿里EB、阿里A和阿里B组成的伪集群，以及由阿里A加上阿里B组成的集群。

从图3中可以看出，随着计算集群中主机数目的增加，完成测试所需要的时间越来越短，这表明计算集群的整体处理能力越来越强。单台盛大超大主机的性能比单台盛大标准主机的性能略好，但比不上由两台盛大标准主机组成的集群。单台盛大超大主机的测试得分比单台盛大标准主机略高，但比不上由两台盛大标准主机组成的集群。由四台盛大标准主机组成的集群，其得分是单台盛大超大主机的三倍。

阿里A型主机与阿里B型主机的性能在同一水平上，虽然阿里B型主机比阿里A型主机多配了1GB的内存，但对性能提升的帮助非常有限。将阿里A型主机与阿里B型主机组成集群后，其性能比单台主机的性能提高一倍。

前面讲过，盛大云超大主机的配置和价格都是标准主机的4倍，换句话说，由4台标准主机组成的集群，在配置和价格上等同于一台超大主机。

性价比

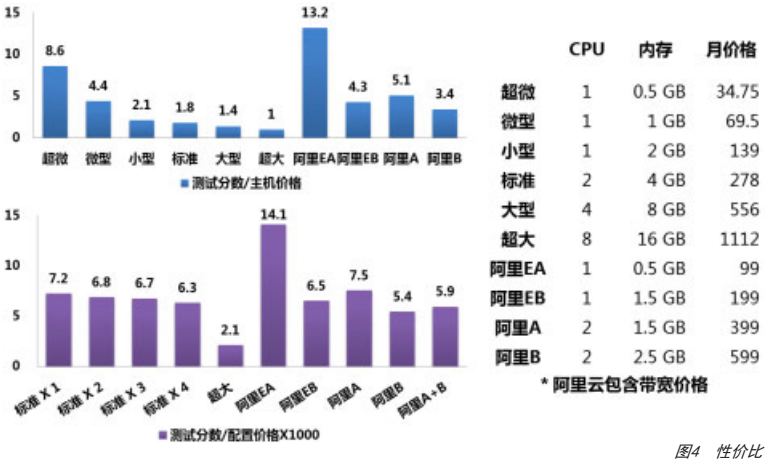
接下来我们比较一下盛大云和阿里云的主机性价比。图4中的上半部分是基于UnixBench测试结果计算的性价比，图4中的下半部分是基于Hadoop wordcount测试结果计算的性价比。在盛大云系列中，如果以UnixBench测试结果来评价，性价比最高的是超微主机，然后依次为微型主机、小型主机、标准主机、大型主机和超大主机。

如果以Hadoop wordcount测试结果来评价的话，由一到四台标准主机组成的集群，其性价比是在同一水平线上的；由一台超大主机组成的伪集群性价比最差，勉强达到其他四个配置的1/3。在阿里云系列中，如果以UnixBench测试结果来评价的话，经济A型主机性价比最高，然后依次是标准A型、经济B型和标准B型主机。如果以Hadoop wordcount测试结果来评价的话，标准A型主机性价比最高，标准A型和标准B型集群次之，标准B型主机性价比最差。

我们还将盛大云的主机和阿里云的主机进行一下横向对比。在UnixBench测试方面，阿里经济A型主机性价比最高，然后依次是盛大超微主机、阿里A型主机、盛大微型主机、阿里经济B型主机和阿里标准B型主机。在Hadoop wordcount测试方面，阿里A型主机的性价比最高，盛大超大主机的性价比最差。盛大标准主机集群比阿里标准B型以及阿里标准A+B集群的性价比略高。

值得一提的是，阿里云的主机价格包含了5MB带宽的价格，盛大云的主机价格不包含带宽价格。如果我们假设5MB带宽的价格为200元的话，那么阿里云主机本身的性价比是全面超过盛大云主机的。

从图4中还可以看出，性价比不是一个独立存在的概念，它与评估的标准有直接关系。用不同的标准（或者应用）来评价同一台主机，得到的结果可能会不同。



	CPU	内存	月价格
超微	1	0.5 GB	34.75
微型	1	1 GB	69.5
小型	1	2 GB	139
标准	2	4 GB	278
大型	4	8 GB	556
超大	8	16 GB	1112
阿里EA	1	0.5 GB	99
阿里EB	1	1.5 GB	199
阿里A	2	1.5 GB	399
阿里B	2	2.5 GB	599

* 阿里云包含带宽价格

图4 性价比

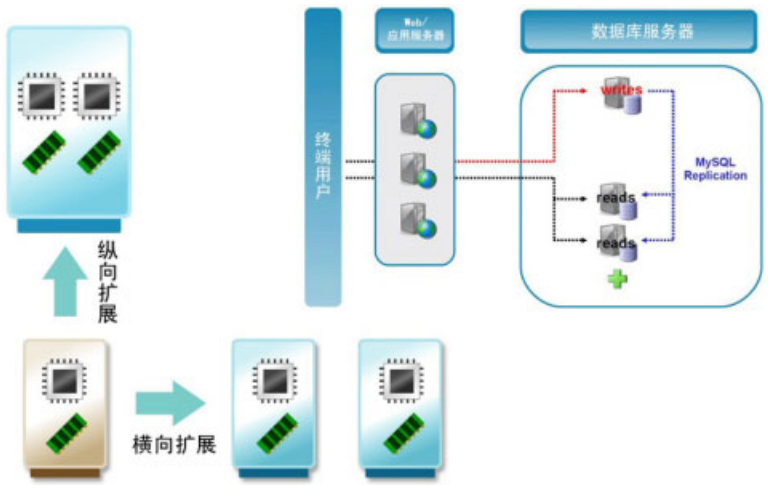


图5 横向扩展VS纵向扩展

此外，在IaaS这个特殊的场景中，可以通过修改超售策略来改变某个产品的性能。因此，同一个产品的价格和性能都是可以随时变化的。如果哪一天盛大云改变了它的定价策略或超售策略，图4中的数据就不再有效了。

横向扩展 vs 纵向扩展

当应用负载出现增长时，我们应该购买更多的标准主机，还是升级到超大主机呢？

横向扩展（Scale Out）和纵向扩展（Scale Up）的目的都是提高系统的整体处理能力。如图5的左下角所示，横向扩展是把计算负载分布到多个节点上，纵向扩展则是给一个节点增加更多的计算资源。在图4中，我们已经看到，由多台标准主机组成的计算集群在性能和性价比两方面都超过超

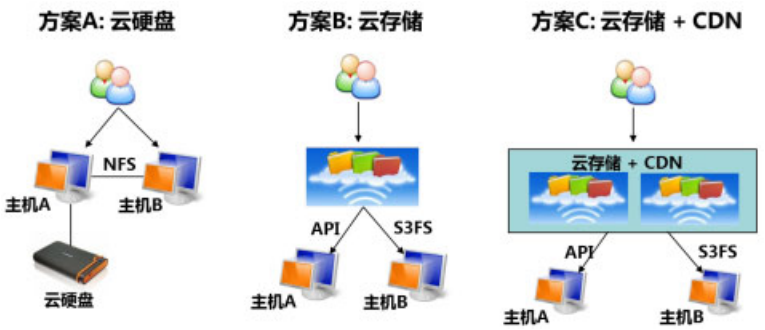


图6 云硬盘VS云存储

大主机。此外，横向扩展还能够给我们带来更多的好处：让系统构架变得清晰，使故障诊断变得容易；避免单点失效，提高了系统的健壮性；可以通过动静分离和读写分离等手段进一步提高系统的性能。因此，当应用需要更强处理能力时，我们应该优先考虑横向扩展而不是纵向扩展。

图5的右上角是一个典型的Web应用横向扩展示意图。做过网站的读者，一定非常熟悉这个架构，本文就不详细解释了。

云硬盘 vs 云存储

经常有人会问：当需要存储空间时，应该选择云硬盘还是云存储？

我们先澄清一下概念。云硬盘是一种虚拟的块设备，可以像U盘一样挂载到云主机上。一块云硬盘在同一时刻只能挂载到一台主机上。如果有多台主机需要同时访问同一块云硬盘，则需要通过NFS等手段进行共享。终端用户需要经过云主机才能够访问到云硬盘上的数据。Amazon所提供的EBS服务，属于我们所说的云硬盘。

云存储则是一种Key/Value数据存储服务，用户通过API进行数据存取操作，也可以通过S3、FS等手段将云存储作为虚拟硬盘同时挂载在多台云主机上。终端用户访问云存储的数据，不需要经过云主机。Amazon提供的S3服务，属于我们所说的云存储。

从云主机的角度来看，理论上云硬盘的终极性能要高于云存储的终极性能。这是因为在云存储上存取数据时还有一些额外开销，这个额外开销在同时存取大量小文件时会比较突出。

从终端用户的角度来看，由于访问云硬盘的数据需要经过云主机，所以云主机的负载就成了影响用户体验的一个不确定因素。如果终端用户访问云存储的数据不需要经过云主机，那么用户体验不会受到云主机的影响。此外，单块云硬盘的容量是受某些物理条件限制的，而云存储几乎可以不受限制地任意扩容。

图6中的方案C是一些厂商正在进行的云存储+CDN的尝试。也就是说，使用云存储存取数据的用户，未来可以自动地享受到CDN服务。当用户访问应用时，静态文件会自动地从离用户最近的地方分发给用户。考虑到云存储技术未来还有很大的发展空间，我认为，除非运行的是需要高速访问磁盘的科学计算型应用，否则还是要尽可能使用云存储来存取数据。

云存储与CDN的融合，是未来基础架构服务领域的一个重要发展方向。建议做基础架构服务的读者多加关注。

总结

在基础架构服务逐渐成为趋势的大前提下，如何有效定义云主机的配置参数，如何在超售的同时保证云主机的性能，如何帮助客户快速创建高性能的计算集群，如何帮助客户实现自动化的横向扩展，如何帮助客户从原来的硬盘存储方案平滑过渡到云存储方案，如何实现云存储与CDN的无缝融合，都是我们团队正在深入研究的内容。



蒋清野

ezCloud联合创始人。曾担任Sun中国技术社区高级经理、天涯社区云计算部门负责人。

责任编辑：杨爽（yangshuang@csdn.net）

腾讯分布式数据仓库解析

文 / 郭玮，李均

TDW（腾讯分布式数据仓库）是在Hadoop和Hive的基础上根据腾讯的业务特点深度定制的，它在稳定性、性能和易用性等方面都有不俗的表现。本文将从应用场景和实践技巧等方面全面解析TDW。

腾讯海量数据处理平台

腾讯是中国拥有最多产品线的互联网公司，它几乎进入了互联网的每一个领域，拥有为数众多的产品。此外，腾讯也是中国拥有最多用户的互联网公司，最新数据显示，腾讯QQ活跃用户已超过7亿。

拥有如此众多的用户和如此庞大的产品线，将不可避免地产生海量数据，如何存储和处理这些数据，如何从这些数据中发掘宝藏，势必成为一个巨大的挑战。图1展示了腾讯海量数据处理平台的概貌。

腾讯海量数据处理平台主要有以下几个特征。

- 所有产品产生的数据通过统一的采集模块进入海量数据处理平台，有实时采集和离线采集两种方式。
- 数据进入平台后，需要实时计算的数据进入流式计算模块，数据及部分流式计算模块的结果数据进入海量存储和批量处理平台——TDW/TDCP（严格地说，TDCP是TDW的一种表现形式）。
- 数据经海量数据处理平台加工后，进入统一分发模块，最终分发给产品，这里提供批量分发、实时分发/查询两种模式。
- 部分经海量数据处理平台加工后的数据将进入BI分析库，供产品做BI分析或使用BI分析工具进行操作。
- 平台内所有模块都在统一数据流工作流管理模块的统一管理下进行工作。

TDW介绍

TDW（Tencent distributed Data Warehouse，腾讯分布式数据仓库）是基于share-nothing架构的分

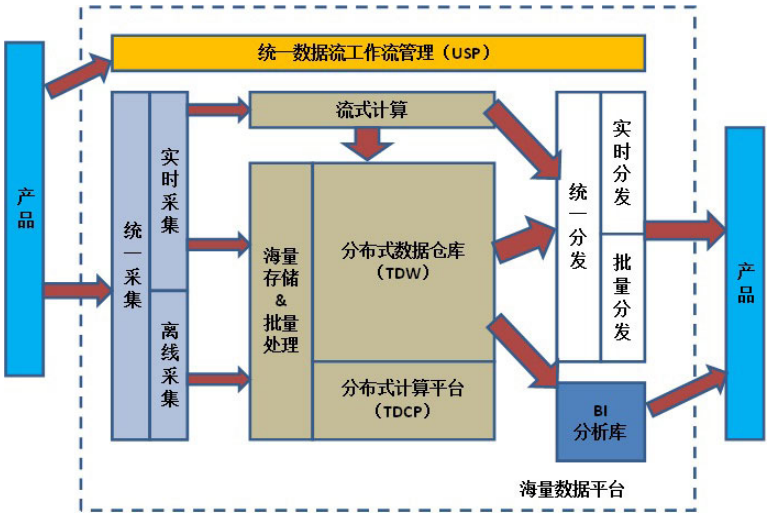


图1　腾讯的海量数据处理平台

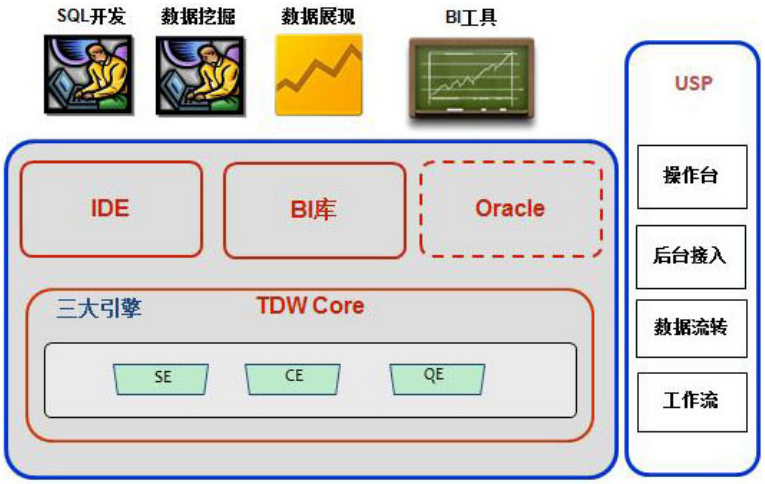


图2 TDW的总体架构

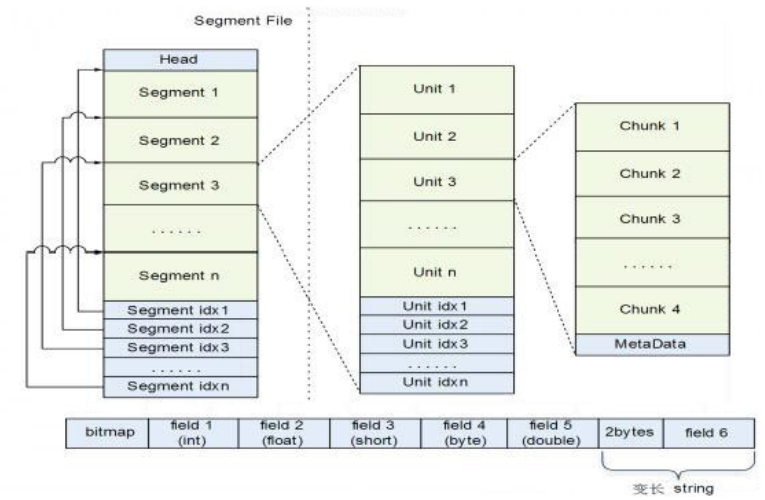


图3 结构化存储后的文件格式

布式处理系统，它是在Hadoop和Hive的基础上根据腾讯的业务特点深度定制的，在分析能力和易用性上得到了全面提升，可用于海量数据的存储和分析。由前面的内容可以看出，TDW是腾讯海量数据处理平台的核心部件。

TDW具有高可用性和高可伸缩性，向用户提供类似SQL的接口，可以提供PB级的存储和TB级的计算能力，用于满足公司日益增长的的海量数据分析需求，帮助发现更多的用户价值。

TDW的总体架构如图2所示，其核心平台（TDW Core）由SE（存储引擎）、CE（计算引擎）和QE（查询引擎）三大引擎组成，分别对应着HDFS、MapReduce和Hive。通过对这三大引擎的深度定制，我们实现了更完善的SQL功能，兼容Oracle函数、过程语言和多种数据格式，以及基于负载感知的公平调度策略，从而使整个系统在稳定性、性能和易用性方面都得到了大幅提升。这些主要的优化工作会在后面的内容中详细描述。

除了Core之外，TDW还包括以下一些辅助模块来共同提供一个完整的海量数据处理方案。

- IDE：开发调试SQL和过程语言的可视化集成开发环境，提供强大的PL和SQL编辑调试环境。
- BI库：提供完善的ODBC接口，与用户的BI展现工具对接，实现BI分析和数据展现。
- USP：统一的调度平台负责数据流和任务流的配置和驱动，同时支持复杂的依赖关系和任务重试报警。

TDW在Hadoop上的实践

如前所述，TDW在Hadoop和Hive的基础上做了大量的改造和优化工作，用于解决实际应用中遇到的难题。下面的内容将详细介绍我们所做的部分优化工作，或者遇到的部分难题及解决方案。

多数据存储格式

结构化存储

由于绝大多数原始的用户数据是文本格式，在进行数据读取时效率较低，所在数据导入过程中，需要将一行文本格式的数据转换成二进制形式存储。对于定长记录，结构化存储可以直接通过记录大小定位到特定的行内容，因此能极大地提升数据访问效率。

结构化存储后的文件格式如图3所示：每个Segment对应HDFS的Block，它是文件切分计算的最小单位。在每个文件的第一个Segment的开始部分增加固定长度的文件内容的描述信息，包括Magic号、数据版本号、是否有变长记录、是否压缩存储、是否加密、加密方式等。描述信息后是具体的文件内容。Head存放各个字段的具体定义（元数据信息），这样每个文件都可以做到自解析，而无须访问额外的元数据系统。存储的二进制数据以Unit为单位，每个Unit的大小尽量均衡，每个Unit是文件读写和压缩的最小单元。Unit里包含很多Chunk，一个Chunk对应一条行记录，Chunk支持可变长的字符串字段存储。

为支持快速按行号定位记录，每一个文件需要

一个索引表，用于快速定位指定的行号数据所在的位置。但为避免存储的开销，不应采用记录索引，而应采用块索引：根据行号找到数据所在的Unit即可。每个文件的最后一个Block存储一级块索引（Segment Index），每个Segment的最后部分存储二级块索引（Unit Index）。

列存储

TDW系统通过引入列存储的概念，减少数据分析过程中不必要的数据访问，减少网络、磁盘I/O，从而进一步提升整体性能。

列存储的文件格式如下。

- 一个表由多个列簇组成（传统的行存储模型可作为一个特例，即所有的字段组成一个列簇）。
- 每个列簇中存储一个或多个字段的值，一个列表的内容存放在一个文件中，文件格式参考前面讲述的结构化存储。
- 为减少计算开销，每个字段值所在的行号保持不变，这样可以很容易地拼出完整的一行。
- 元数据信息基本不会变更。

protocol buffer格式存储

腾讯广告业务统一采用protocol buffer（以下简称为pb）为存储协议，现在其他业务也开始逐步采用。pb有良好的协议扩展能力，描述能力强，解析效率高，相比XML之类的格式可以节省大量的存储空间。因此TDW加入了对pb这种存储格式的支持：我们可以按照pb描述文件创建数据库表，同时支持直接读取pb格式的数据。

用户建表时提供pb数据的定义文件和TDW分析文件，将Message名映射为表名，将Message中的字段映射为这个表的字段，然后在TDW中创建这个表，将这种映射保存到了元数据中。

用户提供protobuf定义文件，TDW解析该文件，将文件中的类型转化为TDW支持的类型，例如将int32转化为int、将int64转化为bigint、将protobuf中的repeated转化为TDW中的array类型。

要使TDW能使用SQL处理protobuf数据，需要FileFormat、SerDe和ObjectInspector三个适配接口。读取流程大致如下。

- FileFormat层从HDFS上的protobuf数据文件中返回序列化后的message的bytesWritable给TDW。
- SerDe将PB数据反序列化为一个message对象。
- ObjectInspector将pb的message解释成TDW能识别的类型。

Hash Join

腾讯90%的数据和计算都是基于用户ID的。针对这种情况，我们实现了Hash分区和Hash Join优化来提高计算的均衡性和性能（如图4所示）。Hash分区是实现计算与存储优化的必要前提。就Join计算而言，实现Hash分区可以将Join计算转换为Map端进行的计算，节省了Reduce阶段的时间消耗。因为相同的Key值（需要与Hash分区的Key一致）必然位于同一个Hash分区中，而且通过控制Hash分区的数目来控制Join计算的粒度，进一步优化对计算资源的利用。

对于已在Hash分区中的表来说，如果是针对非Hash Key（如城市或者频道等字段）做Join计算，需要用Merge Join来实现。具体流程如图4所示：Map阶段扫描表数据，把相同Join Key的记录发送到同一个Reduce，Reduce端按照Join Key排好序后采用Merge Join的算法进行关联计算。与Hash Join相比，Merge Join多了Reduce过程，效率会有所降低。

数据协同摆放

Hash Join通过把在一个Hash分区的数据放到同一个Map上做计算，节省了Reduce时间，从而达到了提升效率的目的。但根据HDFS默认的Block摆放策略，同一Hash分区的数据可能分布在不同的Slave节点上，这样会导致Hash Join时，Map需

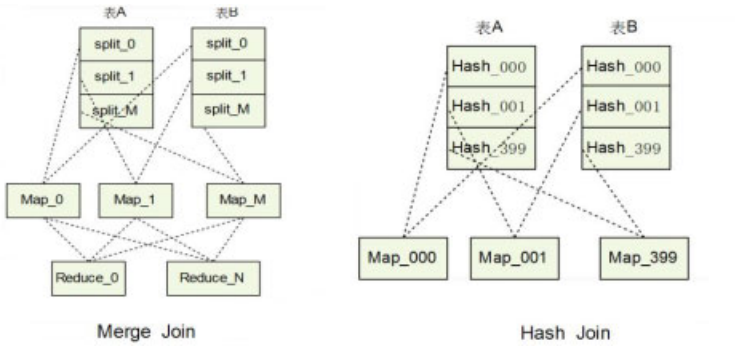


图4 Merge Join和Hash Join

要通过网络I/O把相同Hash分区的数据放到同一节点计算。

数据协同摆放（Colocation）的目的是要修改NameNode的Block摆放策略，使相同Hash分区的数据放在同一个节点。为了更好地适应TDW集群中节点数目的变更，Hash分区数据存放节点的选择将模仿一致性Hash算法（如图5所示）。假设最终确定的Hash分区数为N，则将TDW集群中的机器映射到序号0~N-1，并将IP和序号的映射关系保存到一个配置文件中，由NameNode通过命令读取。在节点中的机器由于批量更新（含上下架）出现变动时，TDW管理员修改该配置文件，并运行hadoop dfsadmin -refreshNodes手动更改。这个命令可以动态读取配置文件中的节点配置，无需重启NameNode。当系统节点配置改变后，管理员应修改配置文件并运行该命令。

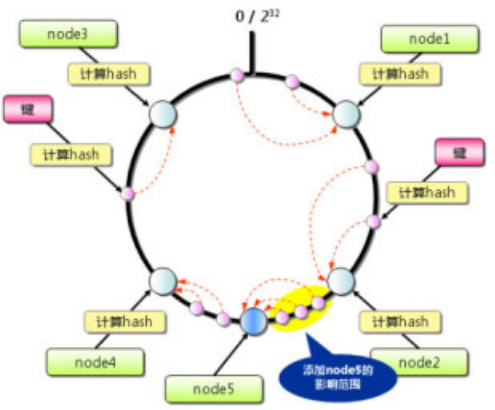


图5　一致性hash算法

基于负载感知的公平调度策略

TDW调度器的资源（Slot、内存、硬盘）是按Pool来管理和分配的。不同Pool之间的共享资源使用配置的分配策略做调度。任务调度到哪个Pool是由该任务user的uesr.name/group.name和任务的service level（online/offline）共同决定的。

Pool的配置可参考图6。

- Key Pool可以帮助运维人员处理特殊情况。某个任务因特殊原因不能在指定时间允许的情况下出结果时，运维人员可以手工将任务提交到Key Pool，调度器会将所有可用资源都优先分配给Key Pool任务使用。
- Online Pool用于线上任务处理，如BU每天要执行的日报、月报等报表功能，通过预留资源的方式保证用户的任务在预定时间可以完成。
- Offline Pool用于线下计算任务，通常为线下的任务如数据挖掘、数据分析等对任务的完成时间没有强制要求的任务。通常不为Offline Pool设置预留资源，在集群任务负载不高时完成任务的计算。Online/Offline Pool都可以设置Pool的资源上限，以保证任务不会消耗过多集群计算资源。



图6　Pool的配置

除了基于Slot来分配任务之外，TDW基于负载感知的调度器还会考虑节点系统资源的实际状况问题。例如，空闲槽的计算节点负载本来就比较重或者计算节点的内存已不足以满足一个task执行时，调度器就会跳过这个节点，以免某些计算节点在自身负载很重的情况下仍被迫接受新任务的状况。

分时计算框架

在TDW上执行的任务大多是以天为周期运行的，这里最大的问题就是需要全天数据准备完整后才能进行分析任务。

TDW分时计算框架的主要目的是充分利用白天时间参与计算，并且和用户的MapReduce程序接口无缝切换。其基本思想与原理如下。

用户数据实时传输到HDFS中，每隔一段时间，平台对用户上传的数据进行Map预处理操作，将其

产生的中间结果存储在HDFS中，并通过一定结构保证数据有序性，以便后续高效的读取。直到在数据完整并且最后一次Map操作结束后，采用与Hadoop兼容的数据读取策略将HDFS的中间结果数据按照Key值顺序依次读出，对每个Key值的关联数据进行Reduce归并操作，得到最终统计结果（如图7所示）。

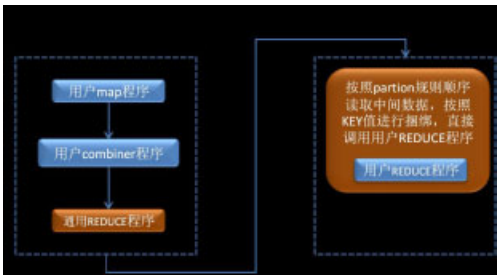


图7　经过Map和Reduce得到最终结果

优化后，平台的Map任务已经分散到全天24小时里，且不会给用户的MapReduce程序带来任何变化，实现了无缝切换。其主要原理就是通过一个通用Reducer程序将用户分时执行Map任务的Key/Value对中间结果进行高效结构化存储，最后一个分时Map任务处理完成后，扩展平台数据读取接口（InputFormat）按照平台缺省数据的读取方式归并各时段中间结果，进行有序读取，最终直接调用用户的Reduce程序归总统计结果。

差异化多级存储

存放在TDW上的数据都会有生命周期的管理。但在生命周期管理基础之外，我们还有差异化压缩存储策略。例如有一些业务数据，存储时间相对来说比较长，有些能够达到一年甚至三年存储时间。但活跃度相对来说较低，被访问到的可能性也比较小，如果还是不做区分地采用这个策略，实际上是种浪费。针对这种情况，我们提供了差异化的压缩算法，它的平均压缩比率比LZO棒得多，因此相对比较冷的数据可以采用差异化压缩存储策略。

此外，我们还提供了差异化副本策略，有1.4和2.6两种选择，在节约存储的同时并没有降低数据安全。实际上还可以选择更激进的策略，例如设置两个副本（缺省是三份副本）。这样我们就有了三种压缩策略：结构化压缩、差异化压缩和差异化副本策略。可针对不同热度的数据采用不同的压缩策略。我们对整个集群中的数据实行差异化存储策略后会得到50%的节省，同时数据计算并没有受到影响。

海量数据处理的“大杀器”

伴随着互联网行业的迅猛发展，腾讯亿万用户打造了一站式的在线生活平台。怎样从用户产生的海量数据中挖掘宝藏，进行更精细化的运营，更好地服务于用户，已成为腾讯业务和产品分析人员面临的一大课题。

TDW在Hadoop和Hive的基础上进行了深度定制，使整个系统在稳定性、性能和易用性方面得到了大幅提升。TDW已成为腾讯内部分析人员处理大数据的“大杀器”，用于每天的海量数据存储和计算。TDW每天承载了数十万张的报表计算和数据分析业务，为腾讯各业务的海量数据处理提供更为强大的技术支撑。



郭玮

2004年加入腾讯，目前为腾讯数据平台部技术总监、腾讯海量数据处理平台负责人，主导研发了腾讯分布式数据仓库（TDW），在互联网海量服务、大数据存储与处理、数据平台等领域拥有丰富的经验。



李均

腾讯数据平台部高级工程师。2005年加入腾讯，现从事分布式存储、分布式计算相关的海量数据处理平台架构和开发工作。

责任编辑：杨爽（yangshuang@csdn.net）

高可用的HDFS架构剖析

文 / 代志远

长期在离线领域占据领头羊地位的Hadoop，如今已开始在在线领域发力，因此HDFS面对严峻的考验。本文将对HDFS的NameNode内核的原理、数据结构及社区中最新的HA解决方案进行剖析，同时给出了社区的架构走向。

熟悉大数据领域的人对Hadoop一定不陌生。Hadoop发展到今天已历经6个年头，很多大数据的处理都依赖于它。长期在离线领域里占据领头羊地位的Hadoop，如今已经在在线领域里发力，前身是Hadoop子工程的HBase也早早从Hadoop项目中剥离出来成为Apache开源基金的顶级项目。伴随NoSQL概念的大热，HBase逐步成为海量数据在线存储和查询的不二选择。现在，Hadoop已成长为一个生态系统，其上生长了很多好产品、好项目，具有直接的商业价值。

分布式文件系统一直是这个体系中底层最核心的框架，在它的基础上生长了MapReduce和HBase等，因此分布式文件系统的高可用性是整个体系中最为重要的一个环节。而且伴随着Hadoop社区的发展，分布式文件系统（HDFS）的高可用性逐渐成为社区的重点关注方向，很长一段时间以来这个话题一直是社区的热点。与此同时，社区的良好技术氛围也促使HDFS在不断地发展。Cloudera、Facebook、Yahoo!等国外公司及百度、阿里巴巴、腾讯、华为等国内公司都为HDFS的发展起到了关键性的推动。

HDFS中的单点问题主要体现在NameNode进程的单点，本文将主要介绍HDFS的NameNode内核的原理和数据结构，社区中最新的HA解决方案，以及社区的架构走向，让更多的技术爱好者了解并更好地促进这个领域的发展。

NameNode内核的原理和数据结构

NameNode中有以下三个关键的数据结构。

- NameSpace
- BlockMap
- LeaseManager

NameSpace

HDFS支持文件路径名，路径名以树形结构表示，完整的路径名信息存储在NameSpace数据结构中。在HDFS中，每个文件划分为若干个Block，每个Block都有唯一的标识叫blockid，并且是全局不重复的。在NameSpace中记录了文件有哪些blockid及blockid的顺序（包括blockSize）。此外，NameSpace中还会记录文件的读写权限及文件的拥有者和用户组信息，以及目录的容量等。在此数据结构中有目录和文件两种结构。

目录的结构为：

```
DIR{
    目录名，
    父目录，
    目录权限，
    目录容量（最大存储容量和最大子节点容量），
}
```

文件结构为：

```
FILE{
    父目录,
    文件权限,
    Block大小,
    默认冗余因子,
    Block列表,
}
```

看到以上数据结构，大家会想到哪些要点？如果现在有两台NameNode，那么我们会考虑到，两者间的数据必须保证强同步，否则一旦发生宕机，用户新创建的文件很可能会丢失。

BlockMap

从NameSpace的数据结构中，我们看到File的数据结构中有一个Block列表结构，那么此时BlockMap数据结构就与Block列表遥相呼应。从HDFS上层看，用户从来不会去关心自己上传到HDFS中的文件究竟被切分了几份，每份都存储在哪些机器上。而在HDFS内部就需要保证每个Block能够存进去、读出来，因此Block的存储地址要记录在某个数据结构中，于是设计了BlockMap数据结构。结构如下：

Block1 -> Block<DN1,DN2,DN3>

Block2 -> Block<DN2,DN5,DN7>

...

注意，这个数据结构并非持久化的存储，而是在内存中构建的对象，是伴随着DataNode心跳信息汇报自身所存储的Block组合而成的。

LeaseManager

客户端的请求写操作文件需要经过NameNode授权，那么在这个体系中就要定义租约的概念，也就是Lease。客户端第一次连接NameNode是为了注册，并对需要请求写操作的文件请求租约，NameNode发现没有其他客户端ID持有这个文件的租约就会自动将这文件租约发给客户端。

从上面的三个结构中，我们可以得出以下结论。

- NameSpace需要持久化。
- BlockMap不需要持久化。
- LeaseManager需要记录。

一直以来，NameSpace的信息都存储在NameNode的FSImage文件中，而写操作信息（包括Lease信息）记录在EditLog中。如果要保持Standby NameNode与Active Node无缝切换，至少在Standby NameNode中要保留实时的NameSpace信息。

解决NameNode的高可用性问题

一直关注社区动态的技术爱好者一定看过Todd Lipcon最近公布的PPT，介绍了HDFS最新的动向和发展重点，其中值得关注的就是HBase的火热发展。而HBase是一个在线存储系统，它所依赖的HDFS需要保持高可用性，但现阶段的HDFS不具备这个高可用条件，因此，社区的走向将从HDFS支持MapReduce转而支持HBase，尤其需要解决的首要问题是高可用性（HA）。与Todd Lipcon的想法类似，我们认为要想解决NameNode的高可用性问题必须具备以下条件。

- Hot
- Client Failover
- Server Failover
- 自动切换时间

Hot&自动切换时间

现阶段，社区对这一点的解决方案是NFS（图1为其结构图）。HDFS用户常用的方式是配置两块元数据存储盘，一块是本地磁盘，另一块是NFS磁盘，这可以保证即使最坏情况下本机彻底宕机，Standby NameNode也可将NFS中存储的元数据恢复到内存中。但对实时性要求较高的HBase来说，恢复时间的长短决定了在线系统的可用性，甚至可能影响交易和网站的访问等。

社区最新的解决方案主要参照了Facebook的AvatarNode，用户甚至可以直接使用AvatarNode代码来取代这部分程序，两者之间有很多共性。Standby NameNode启动后要先从FSImage中恢复NameSpace到内存，而最新的修改就通过EditLog来实现，因为Active NameNode不断地向

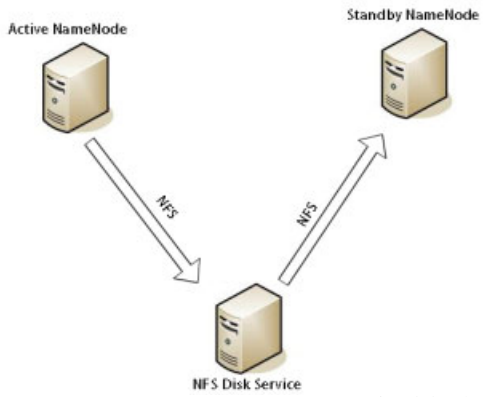


图1 NFS解决方案结构图

EditLog尾部写数据，包括文件（和Lease信息）的新增、删除和修改，那么Standby NameNode就可以异步方式保持与Active NameNode的NameSpace同步。

BlockMap是影响Hot恢复速度的关键因素之一。如何保证仅基于内存存储的BlockMap在Active与Standby之间保持同步呢？社区与AvatarNode给出了很巧妙也很直接的方案——DataNode双写，即同一个DataNode在向NameNode汇报Block信息时，不只汇报给Active Node也要汇报给Standby NameNode，此时Standby NameNode会在内存中自行重建BlockMap的内存结构（大家可自行查阅相关文档，了解更详细的信息）。

Client Failover

相信读者对客户端容灾（Client Failover）很熟悉，这意味着当执行客户端程序（生产程序）时发现原有NameNode连接报错，势必要去寻找Slave NameNode。此时有两个问题：Slave信息是采用相对笨拙的方式以不可更改的方式提供给客户端的，还是一旦发现故障而主动去查询Slave的地址。带着这两个问题，我们有以下思考：采用第一种方法会使容灾方式不够灵活，但却能让客户端直接有效地快速切换；采用第二种方法比较灵活，可以随意增加Slave，但潜在的问题是这个过程要去哪里查询Slave，且这个存储Slave地址的服务器是不是高可用的。思考比较之后，便发现只有ZooKeeper非常适合解决当前的疑虑，但引入ZooKeeper又带来了新的问题——ZooKeeper客户端初次启动时非常慢，而设计HDFS客户端的初衷是轻量级客户端，启动速度快，响应迅速，但引入ZooKeeper之后使得正常命令都要依赖于ZooKeeper的初次连接，与系统初衷相违背。因此最终选择第一种方案。

选择客户端配置Slave地址的方法，需要现有客户端提供retry方式调用RPC，出错后以Slave地址自动重试调用RPC。因此，这个过程自然要对客户是透明的。

Server Failover

故障感知（Failure Detecting）是一个重要课题。在NameNode中有一个关键点，Standby如何获知Active NameNode已出现故障无法提供服务，而将自己的状态切换为Active。这里我们有以下几种假设。

- Standby与Client和Active之间网络不通，但Active与Client之间网络通畅。
- Standby、Client和Active三者之间的网络互不通畅。
- Standby与Client之间网络通，而Standby和Client与Active之间网络不通。
- Active NameNode出现临时故障很快又恢复服务，但已影响到了客户端访问，如Full GC。
- 全网网络瞬断。

上述几种情况就是社区讨论到的Fencing问题，我们有以下解释。

- Standby机器出问题（包括网卡、网口和网线等问题），并且证明Active与Client之间仍旧保持良好的通信关系，这种情况只需要运维人员协助解决问题。
- 出现这种情况的大多数原因是Standby与Active和Client部署在同一个机房，机房的统一出口和各层交换机均出现问题。要避免这种情况，可以将Active与Standby部署到不同机房，也可将Client部署到其他机房。
- 断定Active已经出现问题（可能是硬件层面），这种感知需要上层感应。例如，Standby对Active的心跳，或者将Active的信息存储在高可用的系统中，例如ZooKeeper。
- 确定Active可以切换到Standby，处理这个问题的方式同上。
- 只要将Standby、Client和Active不部署在同一

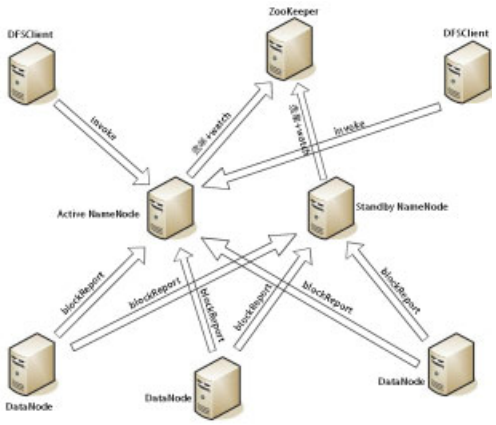


图2 Server Failover解决方案的整体结构

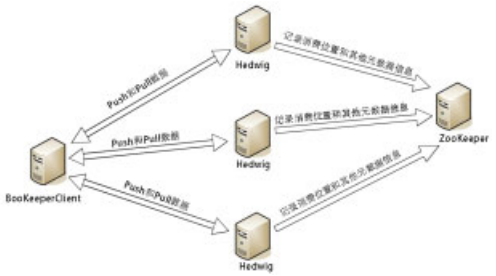


图3 BooKeeper架构

机房，基本就可以避免这一问题。

于是我们考虑到了ZooKeeper，Active将自己的信息注册在ZooKeeper中，Standby监听该信息。保险起见，Standby要增加对Active的心跳，避免出现Standby连不上ZooKeeper但可以连上Active的情况。Hadoop社区的解决方案中考虑了这些因素，并将代码到合并到了trunk中。Server Failover解决方案的整体结构如图2所示。

社区的发展动向

HDFS正在逐渐完善，慢慢使自己更完美。虽然现有的架构是很多人梦寐以求的，但还并不是最完美的。

而在实现HDFS HA时，又会引入了什么问题呢？NameNode性能下降，单次请求延时增加。此时就要分析一下带来性能损耗的原因是什么？答案是NFS，原本写一块本地磁盘，现在要串行写两块磁盘，并且增加一次网络I/O开销，这不得不说是个噩梦，我们是不是能找到更好、更适合的存储来代替这部分的存储呢？

于是社区考虑了多种方案，但我认为最靠谱的方



图4 Region可被若干Hedwig服务器加载

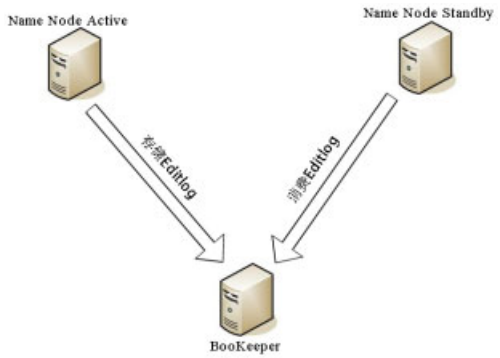


图5 NameNode元数据的共享结构

案是BooKeeper。BooKeeper是基于ZooKeeper实现的分布式内存消息队列，速度快、可用性高，它依靠ZooKeeper来记录元数据（队列被消费的位置等），另外在这个工程中又启动了一个小工程，名字叫Hedwig，这部分就承载了所有来自客户端发送的消息，并且线性拓展能力强。由于可以选择基于内存的存储，效率能得以保证。

BooKeeper架构如图3所示。BooKeeper中有Region概念，用于划分数据服务器区域。一个逻辑上的Region可以被若干任意的Hedwig服务器加载，达到容灾和线性拓展的目的，图4是其结构图。NameNode元数据的共享结构如图5所示。

最后，我想抛出一个问题，当在线系统面临的数据规模达到离线的数据规模，在线系统对NameNode的压力达到现在离线系统对NameNode的压力时，又会出现什么问题呢？答案是元数据庞大，Request处理延时升高，全局锁的致命缺点。要真正解决这个问题恐怕要对NameNode的数据结构做一番优化，并且将全局互斥的数据结构降锁到文件锁（行级锁）。



代志远

从事DFS、MapReduce、Bigtable框架开发多年，有深厚的分布式研发和架构经验，现就职于阿里巴巴集团核心系统部。

责任编辑：杨爽（yangshuang@csdn.net）

Swift：OpenStack对象存储

文 / 程辉

OpenStack Object Storage（Swift）是OpenStack开源云计算项目的子项目之一，被称为对象存储，提供了强大的扩展性、冗余和持久性。本文将从架构、原理和实践等几方面讲述Swift。

Swift并不是文件系统或者实时的数据存储系统，它称为对象存储，用于永久类型的静态数据的长期存储，这些数据可以检索、调整，必要时进行更新。最适合存储的数据类型的例子是虚拟机镜像、图片存储、邮件存储和存档备份。因为没有中心单元或主控结点，Swift提供了更强的扩展性、冗余和持久性。Swift前身是Rackspace Cloud Files项目，随着Rackspace加入到OpenStack社区，于2010年7月贡献给OpenStack，作为该开源项目的一部分。Swift目前的最新版本是OpenStack Essex 1.5.1。

新浪SAE团队对Swift有将近一年的研究和运营经验。在深入剖析Swift架构和原理、完全掌握Swift源码，并且经过一段时间的测试和运营之后，我们决定将推出基于Swift的SAE Storage服务。目前，已完成开发，并于一个月前开始线上运行，且表现非常出色。因此，下面将分享一下我们在Swift上的一些研究和工作。

Swift特性

在OpenStack官网中，列举了Swift的20多个特性，其中最引人关注的是以下几点。

极高的数据持久性

一些朋友经常将数据持久性（Durability）与系统可用性（Availability）两个概念混淆，前者也理解为数据的可靠性，是指数据存储到系统中后，到某一天数据丢失的可能性。例如Amazon S3的数据持久性是11个9，即如果存储1万（4个0）个文件到S3中，1千万（7个0）年之后，可能会丢失其中1个文件。那么Swift能提供多少个9的SLA呢？下文会给出答案。针对Swift在新浪测试环境中的部署，我们从理论上测算过，Swift在5个Zone、5×10个存储节点的环境下，数据复制份是为3，数据持久性的SLA能达到10个9。

完全对称的系统架构

“对称”意味着Swift中各节点可以完全对等，能极大地降低系统维护成本。

无限的可扩展性

这里的扩展性分两方面，一是数据存储容量无限可扩展；二是Swift性能（如QPS、吞吐量等）可线性提升。因为Swift是完全对称的架构，扩容只需简单地新增机器，系统会自动完成数据迁移等工作，使各存储节点重新达到平衡状态。

无单点故障

在互联网业务大规模应用的场景中，存储的单点一直是个难题。例如数据库，一般的HA方法只能做主从，并且“主”一般只有一个；还有一些其他开源存储系统的实现中，元数据信息的存储一直以来是个头痛的地方，一般只能单点存储，而这个单点很容易成为瓶颈，并且一旦这个点出现差异，往往能影响到整个集群，典型的如HDFS。而Swift的元数据存储是完全均匀随机分布的，并且与对象文件存储一样，元数据也会存

储多份。整个Swift集群中，也没有一个角色是单点的，并且在架构和设计上保证无单点业务是有效的。

简单、可依赖

简单体现在架构优美、代码整洁、实现易懂，没有用到一些高深的分布式存储理论，而是很简单的原则。可依赖是指Swift经测试、分析之后，可以放心大胆地将Swift用于最核心的存储业务上，而不用担心Swift捅篓子，因为不管出现任何问题，都能通过日志、阅读代码迅速解决。

应用场景

Swift提供的服务与Amazon S3相同，适用于许多应用场景。最典型的应用是作为网盘类产品的存储引擎，比如Dropbox背后就是使用Amazon S3作为支撑的。在OpenStack中还可以与镜像服务Glance结合，为其存储镜像文件。另外，由于Swift的无限扩展能力，也非常适合用于存储日志文件和数据备份仓库。

Swift架构概述

Swift主要有三个组成部分：Proxy Server、Storage Server和Consistency Server。其架构如图1所示，其中Storage和Consistency服务均允许在Storage Node上。Auth认证服务目前已从Swift中剥离出来，使用OpenStack的认证服务Keystone，目的在于实现统一OpenStack各个项目间的认证管理。

主要组件

Proxy Server

Proxy Server是提供Swift API的服务器进程，负责Swift其余组件间的相互通信。对于每个客户端的请求，它将在Ring中查询Account、Container或Object的位置，并且相应地转发请求。Proxy提供了Rest-full API，并且符合标准的HTTP协议规范，这使得开发者可以快捷构建定制的Client与Swift交互。

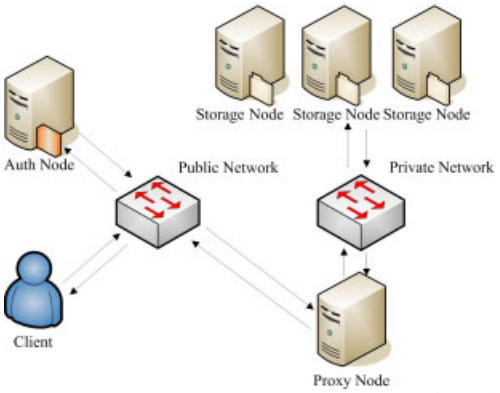


图1 Swift部署架构

Storage Server

Storage Server提供了磁盘设备上的存储服务。在Swift中有三类存储服务器：Account、Container和Object。其中Container服务器负责处理Object的列表，Container服务器并不知道对象存放位置，只知道指定Container里存的哪些Object。这些Object信息以sqlite数据库文件的形式存储。Container服务器也做一些跟踪统计，例如Object的总数、Container的使用情况。

Consistency Servers

在磁盘上存储数据并向外提供Rest-ful API并不是难以解决的问题，最主要的问题在于故障处理。Swift的Consistency Servers的目的是查找并解决由数据损坏和硬件故障引起的错误。主要存在三个Server：Auditor、Updater和Replicator。

Auditor运行在每个Swift服务器的后台持续地扫描磁盘来检测对象、Container和账号的完整性。如果发现数据损坏，Auditor就会将该文件移动到隔离区域，然后由Replicator负责用一个完好的拷贝来替代该数据。图2给出了隔离对象的处理流图。

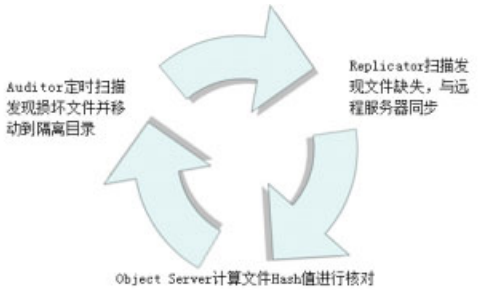


图2 隔离对象的处理流图

在系统高负荷或者发生故障的情况下，Container或账号中的数据不会被立即更新。如果更新失败，该次更新在本地文件系统上会被加入队列，然后Updaters会继续处理这些失败了的更新工作，其中由Account Updater和Container Updater分别负责Account和Object列表的更新。

Replicator的功能是处理数据的存放位置是否正确并且保持数据的合理拷贝数，它的设计目的是Swift服务器在面临如网络中断或者驱动器故障等临时性故障情况时可以保持系统的一致性。

Ring

Ring是Swift最重要的组件，用于记录存储对象与物理位置间的映射关系。在涉及查询Account、Container、Object信息时，就需要查询集群的Ring信息。

Ring使用Zone、Device、Partition和Replica来维护这些映射信息。Ring中每个Partition在集群中都（默认）有3个Replica。每个Partition的位置由Ring来维护，并存储在映射中。Ring文件在系统初始化时创建，之后每次增减存储节点时，需要重新平衡一下Ring文件中的项目，以保证增减节点时，系统因此而发生迁移的文件数量最少。

原理

Swift用到的算法和存储理论并不复杂，主要有几下几个概念。

一致性哈希算法

Swift利用一致性哈希算法构建了一个冗余的可扩展的分布式对象存储集群。Swift采用一致性哈希的主要目的是在改变集群的Node数量时，能够尽可能少地改变已存在Key和Node的映射关系。

该算法的思路分为以下三个步骤。

首先计算每个节点的哈希值，并将其分配到一个0~232的圆环区间上。其次使用相同方法计算存储对象的哈希值，也将其分配到这个圆环上。随后从数据映射到的位置开始顺时针查找，将数据保存到找到的第一个节点上。如果超过232仍然找不到节点，就会保存到第一个节点上。

假设在这个环形哈希空间中存在4台Node，若增加一台Node5，根据算法得出Node5被映射在Node3和Node4之间，那么受影响的将仅是沿Node5逆时针遍历到Node3之间的对象（它们本来映射到Node4上）。其分布如图3所示。

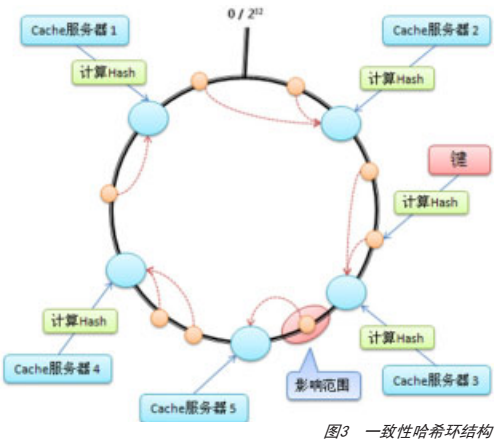


图3　一致性哈希环结构

Replica

如果集群中的数据在本地节点上只有一份，一旦发生故障就可能会造成数据的永久性丢失。因此，需要有冗余的副本来保证数据安全。Swift中引入了Replica的概念，其默认值为3，理论依据主要来源于NWR策略（也叫Quorum协议）。

NWR是一种在分布式存储系统中用于控制一致性级别的策略。在Amazon的Dynamo云存储系统中，使用了NWR来控制一致性。其中，N代表同一份数据的Replica的份数，W是更新一个数据对象时需要确保成功更新的份数；R代表读取一个数据需要读取的Replica的份数。

公式W+R>N，保证某个数据不被两个不同的事务同时读和写；公式W>N/2保证两个事务不能并发写某一个数据。

在分布式系统中，数据的单点是不允许存在的。即线上正常存在的Replica数量为1的情况是非常危险的，因为一旦这个Replica再次出错，就可能发生数据的永久性错误。假如我们把N设置成为2，那么只要有一个存储节点发生损坏，就会有单点的存在，所以N必须大于2。N越高，系统的维护成本和整体成本就越高。工业界通常把N设置为3。例如，对于MySQL主从结构，其NWR数值

分别是N= 2, W = 1, R = 1，没有满足NWR策略。而Swift的N=3, W=2, R=2，完全符合NWR策略，因此Swift系统是可靠的，没有单点故障。

Zone

如果所有的Node都在一个机架或一个机房中，那么一旦发生断电、网络故障等，都将造成用户无法访问。因此需要一种机制对机器的物理位置进行隔离，以满足分区容忍性（CAP理论中的P）。因此，Ring中引入了Zone的概念，把集群的Node分配到每个Zone中。其中同一个Partition的Replica不能同时放在同一个Node上或同一个Zone内。注意，Zone的大小可以根据业务需求和硬件条件自定义，可以是一块磁盘、一台存储服务器，也可以是一个机架甚至一个IDC。

Weight

Ring引入Weight的目的是解决未来添加存储能力更大的Node时，分配到更多的Partition。例如，2TB容量的Node的Partition数为1TB的两倍，那么就可以设置2TB的Weight为200，而1TB的为100。

实例分析

图4中是新浪SAE在测试环境中部署的Swift集群，集群中又分为5个Zone，每个Zone是一台存储服务器，每台服务器上由12块2TB的SATA磁盘组成，只有操作系统安装盘需要RAID，其他盘作为存储节点，不需要RAID。前面提到过，Swift采用完全对称的系统架构，在这个部署案例中得到了很好的体现。图4中每个存储服务器的角色是完全对等的，系统配置完全一样，均安装了所有Swift服务软件包，如Proxy Server、Container Server和Account Server等。上面的负载均衡（Load Balancer）并不属于Swift的软件包，出于安全和性能的考虑，一般会在业务之前挡一层负载均衡设备。当然可以去掉这层代理，让Proxy Server直接接收用户的请求，但这可能不太适合在生产环境中使用。

图4中分别表示了上传文件PUT和下载文件GET请求的数据流，两个请求操作的是同一个对象。

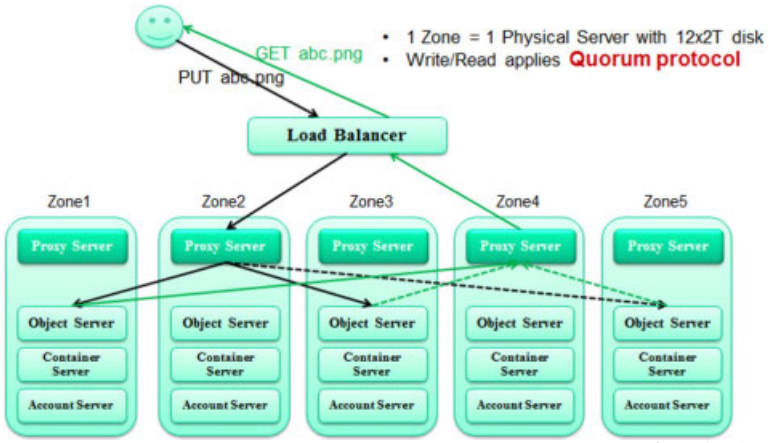


图4 一种Swift部署集群

上传文件时，PUT请求通过负载均衡随机挑选一台Proxy Server，将请求转发到后者，后者通过查询本地的Ring文件，选择3个不同Zone中的后端来存储这个文件，然后同时将该文件向这三个存储节点发送文件。这个过程需要满足NWR策略（Quorum Protocol），即3份存储，写成功的份数必须大于3/2，即必须保证至少2份数据写成功，再给用户返回文件写成功的消息。下载文件时，GET请求也通过负载均衡随机挑选一台Proxy Server，后者上的Ring文件能查询到这个文件存储在哪三个节点中，然后同时去向后端查询，至少有2个存储节点“表示”可以提供该文件，然后Proxy Server从中选择一个节点下载文件。

小结

Swift简单、冗余、可扩展的架构设计保证了它能够用于IaaS的基础服务。在Rackspace Cloud Files服务两年的运行积累使得Swift代码变得越来越成熟，目前已部署在全球各地的公有云、私有云服务中。随着OpenStack的不断完善和发展，Swift将得到更广泛的应用。



程辉

新浪云计算技术经理，曾担任SAE系统开发工程师和运维负责人。他成立了基于OpenStack的项目SWS（Sina Web Services），为新浪内部和合作伙伴提供IaaS服务。

责任编辑：杨爽（yangshuang@csdn.net）

十年磨一剑，开源促创新

英特尔开源技术总监Dirk Hohndel专访

记者 / 董世晓

英特尔开源技术总监Dirk Hohndel（以下简称Dirk），曾担任SuSE公司首席技术官和XFree86项目公司副总裁，自Linux问世以来就是一名积极的开源技术开发者和贡献者。他特别喜欢中国，因此，无论在工作还是生活上，都与中国建立了深厚的感情：2006年即代表英特尔参加首届“开源中国，开源世界”高峰论坛；一直鼓励英特尔开源技术中心的中国团队，大力支持中国开源社区建设；受聘为“中国开源软件推进联盟”智囊团专家，并热心于为中国的开源产业出谋划策；Dirk还收养了一对名叫吉祥和如意的中国小孪生姐妹。

此次，趁Dirk到北京出席第七届“开源中国，开源世界”高峰论坛之际，《程序员》对他进行了专访，请他分享开源方面的洞见。

开源的机遇与挑战

《程序员》：在您看来，开源和闭源的区别有哪些？其中最为突出的是什么？

Dirk：我们坚信，无论是开源还是闭源，都能够创造出有价值的、实用的软件。

开源与闭源的最大区别在于软件开发时应用的方法论不同，因此导致用户对软件使用的世界观也各异：一种用户完全信任在他们系统上运行的软件，相信它没有后门、没有木马、没有间谍软件；另一种用户愿意自己动手处理他们所遇到的问题并希望参与到相关软件的开发中去。

《程序员》：有些人认为，开源一旦被商用，也是需要购买许可证的，所以便说“开源浪得虚名”。您怎样看这个问题？

Dirk：确实，当在商用环境（例如Red Hat Enterprise Linux）中使用开源软件的商用版本时，企业需要付费购买技术支持和维护。但开源软件本身是开放的，任何人、任何公司都能自由使用，例如，即使在商业环境中，Debian Linux或者CentOS也可自由使用。因此，我认为开源还是名副其实的。

《程序员》：在当前，开源面临着怎样的机遇和挑战？

Dirk：在我们身处的这个云计算时代中，开放问题本身已然发生了巨大转变：从我们电脑上运行的软件的开放性，转变为我们托付云服务提供商存储的数据的开放性。因此，开放现在不只是软件的问题，而是事关整个生态系统以及生态系统中的信托和支配关系的问题。这些问题已超出了软件许可证的范畴，不是开源软件本身所能解决的。

英特尔全面支持开源

《程序员》：英特尔的开源理念是什么？开源能为Intel带来什么？



Dirk Hohndel在开源领域是一名老兵

Dirk：很长时间以来，英特尔一直是开源软件的主要支持者和贡献者。在很多重要的项目中，如Linux内核、Tizen和Yocto（嵌入式Linux系统定制环境开源项目）等，英特尔的开源团队都非常活跃。我们认为，开源是加速创新的一种途径，对英特尔在尖端科技方面的创新非常有帮助。

英特尔所有的新技术和新硬件都提供开源软件和闭源软件的支持，而且有时我们会优先提供开源软件支持。例如，英特尔在引入VT（Virtualization Technology）硬件辅助虚拟化技术时，首先在开源软件项目XEN中提供了软件支持。

《程序员》：目前Intel发起和参与的开源项目主要涉及哪些领域？取得了怎样的成果？

Dirk：就像我前面提到的一样，过去十多年以来，英特尔已广泛地参与到开源项目的开发中：从系统内核到工具、库、框架和用户界面组件等，覆盖相当全面。同时，英特尔还倡导了一批重要项目，如Tizen、Yocto等。我们将持续评估新的想法和新的项目，内容涉及数据中心和高性能计算相关的基础设施、移动计算和客户端应用开发等。

对英特尔来说，开源方面的一个重大的里程碑是，在新发布的Linux内核3.4版本中，英特尔成为第一贡献者（计算依据的是两次版本发布期间贡献的补丁数，过去一段时间以来，英特尔一直稳居第二，而这次是英特尔第二次回归第一的位置）。

《程序员》：接下来，英特尔在开源方面会推出哪些举措？

Dirk：抱歉，我不能提前公布我们未来的项目，但可以肯定的是，我们将继续积极参与到开源的各方面中去。

希望更多中国企业和个人投身开源

《程序员》：在中国，开源社区并不如在其他国家那么红火。在您看来，这是什么原因造成的？其他国家有哪些经验值得中国开源界学习？

Dirk：我认为中国是开源软件最大的用户之一。与其他一些国家相比，虽然源自中国的开源项目并不多，但不容忽视的是，中国使用开源的积极性非常高，而且有越来越多的中国企业和个人参与到开源项目的开发中来。

放眼未来，我希望更多的中国企业积极投身于“现有”的开源项目，并能在开源社区中发挥领导作用。开源对我们最大的吸引力之一就是我们可以大声说“这是我们的源码”或者“这是我们的新项目”，但实际上，积极为“现有”开源项目做贡献，却往往蕴含着更大的价值。

《程序员》：您如何看待开源的趋势？

Dirk：可以预见的是，开源软件的使用将得到飞速增长，并将成为IT产业、消费电子及相关领域增长的支柱之一。

Facebook Folly源代码分析(上)

文 / 李凯

Folly是Facebook的一个开源C++11组件库，它提供了类似Boost库和STL的功能，包括散列、字符串、向量、内存分配、位处理等，用于满足大规模高性能的需求。

6月初，Facebook宣布将其内部使用的底层C++组件库Folly开源，本文尝试对Folly库中的几个重要的数据结构代码进行分析，包括一些实现细节的讨论、特点和不足的分析，以及在工程上的应用。本文将首先分析RWSpinlock.h和ThreadLocal.h的源代码。

RWSpinlock.h

顾名思义，RWSpinlock就是使用自旋方式进行临界区资源等待的读写锁，它与pthread_rwlock_t有三个比较重要的区别。

- 通常情况下等待锁的线程不会主动让出CPU，而是循环中不断地尝试获取锁。
- 使用原子操作处理读者计数或写者状态，避免pthread_rwlock_t无论读写的加锁解锁都要在互斥锁的保护下进行。
- 提供类似数据库中”更新锁”的机制，在尽量提供更高并发的情况下，避免死锁。

表1　读写锁状态相容矩阵

	授予类型			
请求类型	NULL	Read	Upgrade	Write
Write	Yes	No	No	No
Upgrade	Yes	Yes	No	No
Read	Yes	Yes	Yes	No

RWSpinlock实现中几处值得关注的地方如下。

自旋锁在锁粒度较小的情况下，使用自选等待方式等锁，可以避免较高的上下文切换代价。而为了自适应多次获取锁失败的情况，可以主动让出CPU。Folly的实现比较简单，硬编码为在自旋1000次后仍无法获取锁的情况下，以后的每次循环都调用sched_yield主动让出CPU以调度其他线程上来运行（要研究更复杂的自适应锁机制，可以参考Solaris内部实现的Adaptive locks或注1提到的论文）。但在获取读锁次数远大于写锁次数的情况下，RWSpinlock的读优先机制可能造成写者的饥饿，而主动让出CPU的机制则可能加重写者的饥饿程度。因此Folly中同时实现了可配置为写者优先的RWTicketSpinLockT锁。

与通常对读计数器加1的思路不同，RWSpinlock使用int32_t的高30位保存读计数，而使用最低两位保存upgrade和write标记。在加解读锁时直接对int32_t的锁状态原子加减0x4，直接避开了对最低两位的修改，执行原子加0x4后再根据原子操作前的最低两位是否有效，来决定是否需要回滚(减

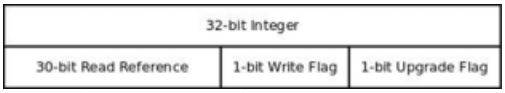


图1　使用比ATOMIC_CAS更高效的ATOMIC_ADD处理读锁的加锁和解锁

0x4)。在多数只获取读锁的情况下，不需要回滚，一次ATOMIC_ADD即完成读锁的加解锁。

Folly中的RWSpinlock还提供了类似数据库中“更新锁”的upgrade锁，用于对锁保护对象先读后改时避免死锁的需求，它与读写锁的状态相容矩阵如表1所示。

Write/Read/Upgrade三种锁状态除了可以和初始状态进行加解锁的双向转换外，也可以在某些锁状态之间进行转换，即原子的释放原来的锁并获取到新的锁，锁状态转换如图2所示。

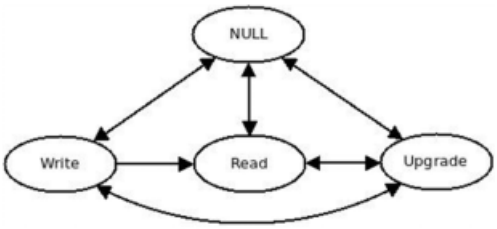


图2 锁状态转换

可以看出，除读写状态外，其他任意两个状态之间都是双向的转换，只有读写之间是单向转换，即在持有写锁的情况下，可以降级为读锁，而在持有读锁的情况下却不能升级为写锁。原因很简单，在两个及以上线程都持有读锁，并尝试获取写锁的情况下，由于释放读锁和获取写锁必须原子性的完成，而要获取写锁就要等待其他线程释放读锁，在这种情况下线程将进入死锁状态。

因此某些对锁保护对象需要先读取再决定是否修改的情况，只能在读取之前就加上写锁，而在读取后需要修改的情况很少时，这种方式代价就比较大，因为它阻塞了其他线程获取读锁。Upgrade就应运而生，从相容性矩阵可以看到，Upgrade锁与读锁相容，而与其他upgrade和写锁不相容，线程在读取数据之前可以先获取upgrade，读取数据之后，如果决定需要修改，就升级为写锁。

一个具体的使用示例如下，如图3所示，考虑实现一个在结点上加锁的B+tree。

查询操作，按照如下操作顺序从根节点开始加锁和查询：

1. 对父节点加读锁；

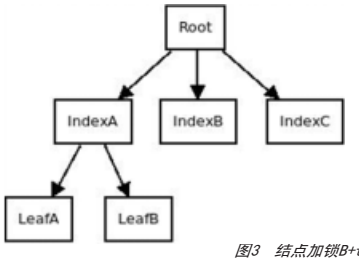


图3 结点加锁B+tree查询

2. 获取子节点的读锁；
3. 释放父节点读锁。

继续在当前节点上执行第二步，直到查询到叶子节点。

更新操作，按照如下操作顺序从根节点开始加锁和查询：

1. 对父节点加upgrade锁；
2. 获取子节点的upgrade锁；
3. 判断子节点如果可能需要分裂或合并，升级父子节点为写锁；
4. 执行子节点分裂或合并，并修改父节点内容；
5. 释放父节点锁；
6. 继续在当前节点上执行第二步，直到叶子节点，对叶子节点执行插入/删除/修改。

因此在B+tree的应用中，由于索引节点的分裂合并操作比较少见，使用upgrade锁，避免与读锁的竞争，只有在必要时才升级为写锁。

关于C++0x中atomic对象中的memory_order，RWSpinlock使用std::atomic保存上面提到的读锁计数、写锁标记、upgrade锁标记，使用了fetch_add、fetch_and、fetch_or、compare_exchange_strong这几个原子操作函数来修改锁状态。作者在不同的场景下使用了三种不同的memory_order，与作者沟通，他的解释如下：

“For example, unlock_shared() can be delayed to other memory_order_release (or memory_order_relaxed), but not memory_order_acquire, which means it ok for the compiler (and machine) to reordering unlock_shared() from different threads.”

但从gcc4.6.3中std::atomic的实现和生成的汇编代码来看，上面提到的几个原子操作函数，直接使用了gcc提供的几个以__sync开头的内置原子

操作，忽略掉了传入的memory_order参数。而只有store函数的行为针对不同的memory_order只有是否增加mfence指令的差别。最后，笔者的建议是在性能影响不大的的情况下，直接使用std::atomic默认的高级别的memory_order，因为通过分析复杂的原子操作指令优化时序，来决定memory_order，收益可能不及它带来的风险。

写者优先的RWTicketSpinLockT锁，提供写锁优先的调度机制，在有线程等待获取写锁的情况下，不再授予读锁，避免在大量加读锁的场景下，写锁很难获取的问题。使用了gcc内置的原子操作__sync_fetch_and_add和__sync_bool_compare_and_swap替代std::atomic，并且也没有用到其他C++0x特性，使用旧版本gcc的项目可以使用这个锁。

Folly注重效率，因此RWTicketSpinLockT中也有几处值得关注的细节。

- 在等待获取锁的自旋中使用pause指令，一方面可以降低CPU的功耗，另一方面还可以帮助CPU优化指令流效率，具体可以参考注2的Intel白皮书。而在写锁不优先的情况下，由于pause带来的延迟可能导致写锁更不容易被获取，因此获取非优先的写锁不使用pause指令。
- 使用SSE并行指令，对多个地址连续的整数一个指令完成++操作。
- 减少分支判断。见源码try_lock_shared()的old.users = old.read。将users与read是否相等的逻辑延迟到CAS操作时顺便判断，尽管在加不上读锁的情况下，要多执行两个自加和一个CAS操作，但在加读锁成功的多数情况下，省去了一次分支判断。
- 使用__sync_fetch_and_add代替__sync_bool_compare_and_swap。RWTicketSpinLockT使用了名为Write、Read、User的三个计数器用来保存读锁计数和写锁标记，方法比较巧妙。读锁需要在user等于read的情况下才可以加上，而写锁则需要满足user等于write，加解锁逻辑如下。

1. 通过对read和user原子加一，获取读锁，同时封锁了加写锁的条件。

2. 通过对read原子加一，获取写锁，同时也就封锁了加读锁的条件；这里通过先对read加一，封锁了后续读锁的条件，然后再等待写锁的条件被满足，实现了写锁优先的逻辑。

3. 通过对write原子加一，释放读锁，同时恢复写锁的加锁条件。

4. 通过对read和write原子加一，释放写锁，同时恢复读锁的加锁条件。

可以看出写锁的获取和读锁的释放可以避免使用CAS，而用一个原子加即可实现。

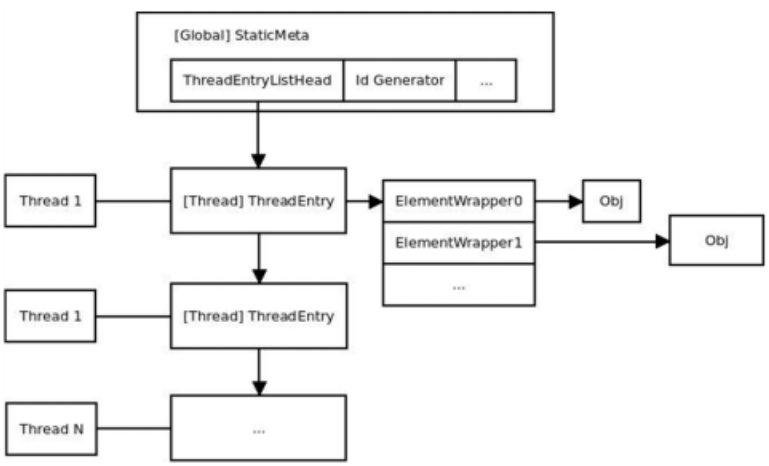


图4 ThreadLocal内存布局

ThreadLocal.h

在服务器编程中，通常会遇到需要为每个线程都分配不同对象的情况，如线程处理一个请求需要使用的临时内存、远程调用需要临时构造的参数等等。在需要的时候临时构造，不仅要付出构造成本，还会有内存申请释放的代价，而使用线程主函数的栈对象，每一层都要传递参数也让代码很不便维护。

Folly中实现的ThreadLocal.h提供了对象的线程局部存储和访问，其功能与pthread_getspecific相似，提供了更方便友好的调用方式，线程退出后自动析构本线程内所欲的私有对象，并且提供遍历所有线程私有对象的接口。实现上使用了GCC

内置特性，实现比pthread库更快的线程私有对象访问。

ThreadLocal内存布局如图4所示，主要由StaticMeta、ThreadEntry和ElementWrapper三者组成。

- StaticMeta为全局唯一结构，主要包括各个线程管理结构组成的链表的头指针和对象ID生成器，用于全局析构和遍历所有线程的私有对象。
- ThreadEntry为线程私有结构，每线程对应一个，主要包括线程线程私有对象的指针数组，管理所有线程私有对象的指针，通过ID获取指定对象的指针。
- ElementWrapper是线程私有对象管理器，每个对象实例对应一个，主要包括指向对象实例的指针和对象的析构方法。

假设要管理的线程私有类型为T，初始化和访问线程私有对象的流程如下。

1. ThreadLocalPtr对象构造时即从StaticMeta为它管理的类型申请了唯一ID。

2. 使用ThreadLocalPtr::get方法通过ID从ThreadEntry管理的ElementWrapper数组中获取一个ElementWrapper对象。

3. 如果ElementWrapper中T的指针为空，则构造一个T的对象，指针和析构方法保存在ElementWrapper对象中。

4. 如果ElementWrapper中的T指针不为空，则直接返回。

在Folly的实现中值得注意的是：ThreadEntry对象在每个线程中有一个，使用gcc内置的static __thread方式声明ThreadEntry，即可实现同一个名字在不同线程访问到的是不同对象，需要注意的是这种方式仅适用于POD类型。由于直接访问对象，这种方式比调用pthread库的pthread_getspecific函数调用方式效率要更高。

但由于ThreadEntry是POD类型，在线程退出时不能自动析构释放它管理的线程私有对象，因此在StaticMeta构造时会申请一个pthread_key_t注册线程退出时的回调函数，在回调函数中遍历当前线程ThreadEntry管理的所有私有对象，依次调用它们的析构方法。因此在Folly的实现中，对pthread库函数仅仅使用了它在线程退出时调用回调函数的功能。

此外还有两处细节值得借鉴。

- ThreadLocal还区分了单个线程退出和整体析构情况下，传给析构方法不同的参数，以便用户在必要的情况下区别这两种情况下析构方法的实现。
- 提供了指定立即析构释放当前线程私有对象的方法，而不必等待线程退出时才释放，这一点在单元测试多个case的情况下可能会使测试变得比较方便。

注释

注1: Adaptive Locks: Combining Transactions and Locks for Efficient Concurrency

注2: Using Spin-Loops on Intel® Pentium® 4 Processor and Intel® XeonTM Processor

注3: A Provably Correct Scalable Concurrent Skip

注4: Hazard Pointers: Safe Memory Reclamation for Lock-Free Objects

注5: Doug Lea. ConcurrentSkipListMap. In java.util.concurrent



李凯

现任淘宝核心系统部存储组技术专家，花名郁白。2007-2010年曾参与百度分布式文件系统研发，2010年至今参与淘宝Oceanbase项目研发。

责任编辑：卢鹏翔（ludx@csdn.net）

用C++进行函数式编程

文 / John Carmack 译 / 王江平

《Quake》作者Carmack认为追求函数式的程序设计有着实实在在的价值，然而，劝说所有程序员抛弃他们的C++编译器，转而启用Lisp、Haskell，或者干脆说任何其他边缘语言，都是不负责任的。

或许本文的每位读者都听说过，当初“函数式编程”（Functional Programming）肩负着为软件开发带来福祉的期望来到这个世界，大家可能还听说过有人将它奉为软件开发的银弹。然而，上维基百科查看更多信息却让人大倒胃口，一上来就引用λ演算和形式系统。很难一眼看出这跟编写更好的软件有什么关系。

我的实效性总结：软件开发中的大部分问题都缘于程序员没有完全理解程序执行中所有可能的状态。在多线程环境中，这一理解的缺失以及它所导致的问题变得更加严重，如果你留意这些问题，会发现它几乎严重到令人恐慌的地步。通过函数式的风格编写程序，可以将状态清晰地呈现给你的代码，从而使代码的逻辑更易于推理，而在纯粹的函数式系统中，这更使得线程竞争条件成为不可能的事情。

我确实相信追求函数式的程序设计有着实实在在的价值，然而劝说所有程序员抛弃C++编译器，转而启用Lisp、Haskell，或者干脆说任何其他边缘语言，那是不负责任的。让语言设计者永远懊恼的是，总会有大量的外在因素压跨一门语言的好处，相对大多数领域来说，游戏开发尤其如此。除了大家都要面对的遗留代码库和有限的人力资源问题之外，我们还有跨平台问题、私有工具链、证书网关、需要授权的技术，以及严酷的性能要求。

如果你的工作环境中可以用非主流语言完成主要开发任务，那应该为你欢呼，不过也等着打板子吧，罪名是项目进展方面的。而对所有其他人：不论你用何种语言工作，通过函数式的风格编写程序都会带来好处。任何时候，只要方便，就应当这么做；而不方便时，也应当仔细想想自己的决定。以后，只要愿意，你可以学学lambda、monad、currying、在无限集上合成懒惰式求值的函数，以及显式面向函数式语言的所有其他方面。

C++语言并不鼓励函数式程序设计，但它也不妨碍你这么做，而且为你保留了深入下层、运用SIMD内在函数基于内存映射文件直接布局数据的能力，或任何其他你发现自己用得着的精华特性。

纯函数

纯函数是这样一种函数：它只会查看传进来的参数，它的全部行为就是返回基于参数计算出的一个或多个值。它没有逻辑副作用。这当然只是一种抽象；在CPU层面，每个函数都是有副作用的，多数函数在堆的层面上就有副作用，但这一抽象仍然有价值。

纯函数不查看也不更新全局状态，不维护内部状态，不执行任何I/O操作，也不更改任何输入参数。最好不要传递任何无关的数据给它——如果传一个allMyGlobals指针进来，这一目标就基本破灭了。

纯函数有许多良好的属性。

- **线程安全** 使用值参数的纯函数是彻底线程安全的。使用引用或指针参数的话，就算是const的，你也应当知晓一个执行非纯操作的线程可能更改

或释放其数据的风险。但即便是这种情况，纯函数仍不失为编写安全多线程代码的利器。

你可以轻松地将一个纯函数替换为并行实现，或者运行多种实现并比较结果。这让代码的试验和演化都更加便利。

- **可复用性** 移植一个纯函数到新的环境要容易很多。类型定义和所有被调用的其他纯函数仍然需要处理，但不会有滚雪球效应。有多少次，你明明知道另一个系统有代码可以实现你的需要，但要把它从所有对系统环境的假设中解脱出来，还不如重写一遍来得容易？

- **可测试性** 纯函数具有引用透明性（referential transparency），也就是说，不论何时调用它，对于同一组参数它永远给出同样的结果，这使它跟那些与其他系统相互交织的东西比起来更易于使用。在编写测试代码的问题上，我从来没有特别尽责；太多代码与大量系统交互，以至于使用它们需要相当精细的控制，而我常常能够说服自己（也许不正确）这样的付出并不值得。纯函数很容易测试，其测试代码就像直接从教科书上摘抄下来的一样：构造一些输入并查看结果。每次遇到一小段目前看起来有些奇技淫巧的代码，我都会把它拆成一个单独的纯函数并编写测试。可怕的是，我常常发现这样的代码中存在问题，意味着我撒下的测试安全网还不够大。

- **可理解性与可维护性** 输入和输出的限制使得纯函数在需要时更易于重新学习，由于文档不足而隐藏了外部信息的情况也会更少。

形式系统和软件的自动推理将来会越来越重要。静态代码分析今天已经很重要了，将代码转换成更加函数式的风格有助于工具对它的分析，或者至少能让速度更快的局部工具所覆盖的问题跟速度慢且更加昂贵的全局工具一样多。我们这个行业讲的是“把事情做出来”，我还看不到关于整个程序“正确性”的形式证明能成为切实的目标，但能够证明代码的特定部分不存在特定种类的问题也是很有价值的。我们可以在开发过程中多运用一些科学和数学成果。

正在修编程导论课的同学可能一边挠头一边想：“不是所有的程序都要这么写吗？”现实情况却是“大泥球”（Big Balls of Mud）程序多，架构清晰的程序少。传统的命令式编程语言为你提供了安全舱口，结果它们就总是被使用。如果你只是写一些用一下就扔掉的代码，那就怎么方便怎么来，用到全局状态也是常事。如果你在编写一年之后仍将使用的代码，那就要将眼前的便利因素跟日后不可避免的麻烦平衡一下了。大部分程序员都不擅长预测日后改动代码将会导致的各种痛苦。

“纯粹性”实践

并非所有东西都可以是纯的，除非程序只操作自己的代码，否则到某个点总要与外部世界交互。尝试最大限度地推进代码的纯粹性可以带来难以想象的乐趣，然而，要达到一个务实的临界点，我们需要承认副作用到某一刻是必要的，然后有效地管理它们。

即使对某个特定的函数而言，这都不是一个“要么全有要么全无”的目标。随着一个函数的纯度不断提高，其价值可以连续增大，而且从“几乎纯粹”到“完全纯粹”带来的价值要低于从“意大利面条状态”到“基本纯粹”带来的价值。只要让函数朝着纯粹的目标前进，即使不能达到完全的纯度，也能改善你的代码。增减全局计数器或检查一个全局调试标志的函数是不纯的，但如果那是它唯一的不足，它仍然可以收获函数式的大部分好处。

避免在更大的上下文中造成最坏的结果通常比在有限的情形中达到完美状态更加重要。考虑一下你曾经对付过的最令人不爽的函数或系统，那种只有全副武装才能应付的，几乎可以确定，其中必有复杂的状态网络和代码行为所依赖的各种假设，而这些复杂性还不只发生在参数上。在这些方面强化一下约束，或至少努力防止更多的代码陷入类似的混乱局面，带来的影响将比挤压几个底层的数学函数大得多。

朝着纯粹性的目标重构代码，这一过程通常包含将计算从它所运行的环境中解脱出来，这几乎必然意味着更多的参数传递。似乎有点奇特——编程语言中的烦琐累赘已被人骂够了，而函数式编程却常常与代码体积的减少相关。函数式编程语言写的程序会比命令式语言的实现更加简洁，其

中的因素与使用纯函数在很大程度上是正交的，这些因素包括垃圾回收、强大的内建类型、模式匹配、列表推导、函数合成以及各种语法糖等。程序体积的减少多半与函数式无关，某些命令式语言也能带来同样的效果。

如果你必须给一个函数传递十多个参数，恼火是应该的，你可以通过一些降低参数复杂性的方法来重构代码。

C++中没有任何维护函数纯粹性的语言支持，这确实不太理想。如果有人通过一些不好的方法把一个大量使用的基础函数变得不再纯粹，所有使用这一函数的代码便统统失去了纯粹性。从形式系统的角度听起来这是灾难性的，但还是那句话，这并不是一念之恶便与佛无缘的那种“要么全有要么全无”的主张。很遗憾，大规模软件开发中的问题只能是统计意义上的。

看来未来的C/C++语言标准很有必要增加一个“pure”关键字。C++中已经有了一个近似的关键字const——一个支持编译时检查程序员意图的可选修饰符，加上它对代码百利而无一害。D语言倒是提供了一个“pure”关键字：http://www.d-programming-language.org/function.html。注意它们对弱纯粹性和强纯粹性的区分——要达到强纯粹，输入参数中的引用或指针需要使用const修饰。

从某些方面来看，语言关键字过于严格了——一个函数即使调用了非纯粹的函数也仍然可以是纯粹的，只要副作用不逃出函数之外即可。如果一个程序只处理命令行参数而不操作随机的文件系统状态，那么整个程序都可看做纯粹的函数式单元。

面向对象程序设计

Michael Feathers（twitter @mfeathers）说：OO通过把移动的部件封装起来使代码可理解。FP通过把移动的部件减到最少使代码可理解。

“移动的部件”就是更改中的状态。通知一个对象改变自己，这是面向对象编程基础教材的第一课，在大多数程序员的观念中根深蒂固，但它却是一种反函数式的行为。将函数和它们操作的数据结构组织在一起，这一基本的OOP思想显然有其价值，但如果想在自己的部分代码中获得函数式编程的好处，那么在这些部分，你必须疏远一下某些面向对象的行为。

无法声明为const的类方法从定义上就是不纯的，因为它们要修改对象的部分或全部状态集合，这一集合可能十分庞大。它们也不是线程安全的，这里戳一下，那里捅一下，一点一点地把对象置成了非预期的状态，这种力量才真正是Bug的不竭之源。如果不考虑那个隐含的const this指针，从技术角度const对象方法仍可看做纯函数，但许多对象十分庞大，大到它本身就足以构成一种全局状态，从而弱化了纯函数的在简洁清晰上的一些好处。构造函数也可以是纯函数，通常应该努力使之成为纯函数——它们接受参数并返回一个对象。

从灵活编程的层面来看，你常常可以用更加函数式的方法使用对象，但可能需要一点接口上的改变。在id Software，我们曾有十年时间在使用一个idVec3类，它只有一个改变自己的void Normalize()方法，却没有相应的idVec3 Normalized() const方法。许多字符串方法也是以类似的方式定义的，它们操作自身，而不是返回执行过相应操作的一个新的副本——比如ToLowerCase()、StripFileExtension()等。

性能影响

在任何情况下，直接修改内存块几乎都是无法逾越的最优方案，而不这么做就难免牺牲性能。多数时候这只有理论上的好处，我们一向都在用性能换生产率。

使用纯函数编程会导致更多的数据复制，出于性能方面的考虑，某些情况下这显然会成为不正确的实现策略。举个极端的例子，你可以写一个纯函数的DrawTriangle()，接受一个帧缓存（framebuffer）参数并返回一个全新的画上三角形的帧缓存作为结果。可别这么做。

按值返回一切结果是自然的函数式编程风格，然而总是依靠编译器实施返回值优化会对性能造成危害，因此对于函数输出的复杂数据结构，传递引用参数常常是合理的，但这么也有不好的一面：它阻止你将返回值声明为const以避免多次赋值。

很多时候人们都有强烈的欲望去更新传入的复杂结构中的某个值，而不是复制一份副本并返回修改后的版本，但这样等于舍弃了线程安全保障，因此不要轻易这么做。列表的产生倒是一种可以考虑就地更新的合理情形。往列表中追加新的元素，纯函数式的做法是返回尾端包含新元素的一个全新列表副本，原先的列表则保持不变。真正的函数式语言都在实现上运用了特别手法，从而使这种行为的后果没有听上去那么糟糕，但如果在典型的C++容器上这么做，那你就死定了。

一项重要的缓解因素是，如今性能意味着并行程序设计，相比单线程环境，并行程序即使在性能最优的情形中也需要更多的复制与合并操作，因此复制造成的损失减少了，而复杂性的降低和正确性的提高这两方面的好处相应增加了。例如，当开始考虑并行地运行一个游戏世界中的所有角色时，你就会渐渐明白，用面向对象的方法来更新对象，这在并行环境中难度很大。或许所有对象都引用了世界状态的一个只读版本，而在一帧结束时却复制了更新后的版本……嗨，等一下……

如何行动

在自己的代码库中检查某些有一定复杂度的函数，跟踪它能触及的每一比特外部状态以及所有可能的状态更新。即使对它不做一点改动，把这些信息放入一个注释块就已经是极好的文档了。如果函数能够——比方说，通过渲染系统触发一次屏幕刷新，你就可以直接把手举在空中，声明这个函数所有的正副作用已经超出了人类的理解力。

你要着手的下一项任务是基于实际执行的计算从头开始重新考虑这个函数。收集所有的输入，把它传给一个纯函数，然后接收结果并做相应处理。

调试代码的时候，让自己着重了解那些更新的状态和隐藏的参数悄然登场，从而掩盖实际动作的部分。

修改一些工具对象的代码，让函数返回新的副本而不是修改自身，除了迭代器，试着在自己使用的每个变量之前都加上const。

（感谢John Carmack和Mike Acton对本文的授权，原文链接为http://www.altdevblogaday.com/2012/04/26/functional-programming-in-c/）



John Carmack

享誉世界的著名程序员，id Software创始人之一。Doom和Quake系列游戏作者。

责任编辑：卢鹏翔（ludx@csdn.net）

寻找机遇 创造未来 CSDN 热门职位全新推荐
■详细信息请参见CSDN网站

北京法国电信研发中心

Software engineer in the web service system development

orange™

Position requirements:

- Master degree in telecommunication, Compute Science OR equivalent
- Experience in J2EE, SOA, Spring, Hibernate, Struts, XML, REST, HTML5 development
- Experience in cloud computing development, such as SaaS, PaaS, etc.
- Strong programming skills, at least 3+ years experience in Java software development
- Familiar with the software development methodology
- Familiar with networking technology
- Good English communication skill
- Good team working spirit and communication skills, and hard working as well

简历投递邮箱：hr@orange-ftgroup.com.cn

CSDN

CSDN 成立于 1999 年，专注 IT 信息传播、技术交流、教育培训和专业技术人才服务。目前拥有超过 2100 万注册用户，覆盖 90% 以上的开发者和 IT 专业族群，以及 90% 以上的大中型 IT 相关企业，CSDN 致力于为开发者及 IT 专业人士创造价值。

CSDN 全球最大中文IT社区

现诚聘如下职位：

- UI 设计师
- PHP 开发工程师
- Ruby 开发工程师
- 问答社区编辑
- 问答社区运营
- 移动频道编辑
- 软件研发频道编辑
- 业界报道记者

简历投递邮箱：hrzhaopin@csdn.net
网址：www.csdn.net

寻找机遇 创造未来 CSDN 热门职位全新推荐

■详细信息请参见CSDN网站

深圳市明源软件股份有限公司

深圳市明源软件公司，始终专注于房地产行业信息化领域，是目前国内领先的房地产应用软件及解决方案供应商。

Mysoft 明源®

现诚聘如下职位：

- iPhone 开发工程师
- .NET 开发工程师
- 软件测试工程师
- 软件设计 / 需求分析工程师
- 高级开发工程师
- 项目经理

地址：深圳市南山区高新科技园南区高新南一道中科大厦
简历投递邮箱：hr@mysoft.com.cn
网址：http://www.mysoft.com.cn

广州多益网络科技有限公司

多益网络，创立于 2006 年 7 月，专注于网络游戏研发与运营，拥有国内位居行业前列的研发运营团队及一流的产品研发能力，是广州领先的专业网游研发及运营公司。

多益网络 www.henhaoji.com

现诚聘如下职位：

- 游戏服务端程序员
- C++ 软件工程师
- 游戏客户端程序员
- 网站技术经理
- C++ 高级程序员
- asp.net 高级程序员

地址：广州市天河区翰景路 1 号金星大厦 3-7 楼
简历投递网址：hr.henhaoji.com
网址：www.henhaoji.com

上海交通大学E-learning Lab

上海交通大学 E-learning 实验室是上海交通大学优秀的独立研究中心。开展各种领先性技术研究，旨在推进 E-learning 教学领域内的理论、技术及产业化发展。

现诚聘如下职位：

- 高级软件研发工程师
- 视频编解码技术研发工程师
- 高级 Web 应用研发工程师

地址：上海市华山路 1954 号交通大学浩然高科技大厦 6 楼
简历投递邮箱：viena@sjtu.edu.cn
网址：dlc.sjtu.edu.cn

北京中油瑞飞信息技术有限责任公司

中国石油集团旗下专业 IT 技术服务供应商。2006 年成立至今已形成软件服务、ERP 咨询、财务金融、油田应用、海外业务、系统集成、信息安全、数据中心八大业务板块，员工 2200 余人，拥有信息系统集成二级、CMMI 三级和 ISO20000 等多项国际资质。

招聘职位：

- 系统工程师（系统运维、监控实施）
- 云计算系统工程师（软件、虚拟化、网络、存储、服务器、综合、安全、核心技术、综合）
- 软件工程师（Java、.NET、BI 产品研发）

简历投递邮箱：richfit_hr@cnpc.com.cn 请以“应聘职位 + 姓名”命名
电话：010-59981054、59981057 网址：www.richfit.com

广州瀚信通信科技股份有限公司

HANTELE 瀚信

现诚聘如下职位：

- C/C++ 中级开发工程师
- C# .NET 中级开发工程师
- C/C++ 高级开发工程师
- C# .NET 高级开发工程师
- C/C++ 系统架构师
- C# .NET 系统架构师
- C# .NET 初级开发工程师
- 研发项目经理（C/C++ 或 .NET、Delphi 等均可）

简历投递邮箱：yfhr@hantele.com，请以“职位 + 姓名 +CSDN”作为邮件标题

聚胜万合信息技术（上海）有限公司

聚胜万合创建于 2009 年初，并在当年获得美国光速创投基金的首轮融资。公司聚合了业界优秀的互联网技术、广告营销、互动创意专家，是专业从事精准营销及数字营销的专业广告技术和服务机构。

MediaV

现诚聘以下职位（工作地点：上海）：

- 云计算工程师
- 资深系统运维工程师
- 前端开发工程师 /PHP 软件工程师
- DBA
- 前端开发工程师（CSS+JavaScript）

地址：上海市闸北区恒丰路 568 号 恒汇国际大厦 10 楼
简历投递邮箱：hr@mediav.com 网址：www.mediav.com

网易有道

作为网易自主研发的全新中文搜索引擎，有道搜索致力于为互联网用户提供更快更好的中文搜索服务。依托网易强大的产品服务平台和丰富的资源优势，有道搜索吸纳了众多优秀的创新人才，并正在高速发展中。

现诚聘如下职位：

- 前端开发工程师
- 测试开发工程师
- Web 开发工程师
- 研发工程师
- 有道笔记 / 有道词典 UI 设计师

北京市海淀区中关村东路 1 号清华科技园 3 号楼（创业大厦）2 层
简历投递邮箱：wangjian@rd.netease.com 网址：www.youdao.com

上海瑞创网络科技股份有限公司

2006 年 1 月成立于上海浦东，是一家以互联网运营和软件开发为一体的知名互联网企业。公司成立七年来，每年都以 100% 的速度增长，业务发展非常迅速。2011 年收入超过 3 亿元，利润超过 1 亿元，纳税 5000 万元，跻身国内知名互联网公司行列。

现诚聘以下职位：

- 高级 PHP 开发工程师
- iOS 开发工程师
- C++ 开发经理
- 测试工程师
- Flash 开发工程师

地址：上海浦东新区亮秀路 112 号 Y2 座 9 楼
简历投递邮箱：hr@ruichuang.net 网址：www.ruichuang.net

上海征途信息技术有限公司

上海巨人网络科技有限公司成立于 2004 年 11 月 18 日，是以网络游戏为发展起点，集研发、运营、销售为一体的互动娱乐企业。

现诚聘如下职位：

- 资深客户端软件工程师
- 资深服务器端软件工程师
- 资深 3D 引擎软件工程师
- 资深 Java 游戏开发工程师
- 资深 Flash 游戏开发工程师
- Web 前台开发工程师
- 资深 Web 开发工程师（PHP）
- Java 软件开发工程师

地址：上海市徐汇区宜山路 700 号 3 号楼
简历投递邮箱：hr@ztgame.com 网址：hr.ztgame.com

ThoughtWorks

如果你**热爱代码如生命**，喜欢有事没事就记录和分享自己的思考与实践，希望远离办公室政治，追求公平合理，追求成为一名优秀的程序猿、攻城狮。那不容置疑 ThoughtWorks 绝对适合你！
你将和一群 Geek **在编码中一较高下**！
你将和优秀**的咨询师**一起，享受在帮助客户解决棘手问题之后的成就感！
你将在不同国家、不同项目、不同技术中**快速成长**！
工作地点：**北京、西安、成都、上海**

现诚聘以下职位：

- Java/C#/Ruby 开发工程师
- 质量保证工程
- 前端开发工程师

简历投递邮箱：job@thoughtworks.com

DeNA 中国

DeNA 是日本无线互联网的龙头企业，2011 第四季度营收 4.45 亿美元。DeNA 中国扎根本土，主要业务为智能手机平台“mobage 梦宝谷”的开发及运营；第三方开发者平台的开发及运营；海外游戏的本地化及中国游戏的海外发行等。

现诚聘如下职位：

- 游戏服务器工程师（PHP/Perl/Ruby）
- 资深 iOS 工程师
- Web 后台开发工程师（Java/Perl）
- 技术支持工程师
- 项目经理
- 数据库开发工程师（MySQL）

简历投递邮箱：hr@denachina.com，并在邮件标题上注明“程序员杂志” 更多职位请至 http://job.denachina.com

北京格致璞科技有限公司

北京格致璞科技有限公司（简称：Gozap）凭借自身的实力与长期积累的经验，以用户为中心，本着诚信的原则，为广大互联网及无线互联网用户提供优秀的产品和服务。

现诚聘如下职位：

- Flex/Flash/AS3 开发工程师
- 测试工程师
- 手机软件开发工程师（Android/iPhone/WP7）
- 前端开发工程师
- Java 开发工程师
- 后台开发工程师
- C/C++ 工程师
- 系统运维工程师
- Python/PHP/HTML5 开发

地址：北京市西城区西直门西环广场 T1-1102
简历投递邮箱：hr@gozap.com 网址：www.gozap.com

新书上架

本月新书



推荐系统实战

作者：项亮

本书通过大量代码和图表全面系统地阐述了和推荐系统有关的理论基础，介绍了评价推荐系统优劣的各种标准和方法，总结了当今互联网领域中各种和推荐有关的产品和服务。



Go语言·云动力

作者：樊虹剑

本书是Go语言程序设计入门书，介绍了静态类型、流程控制、函数、动态类型、面向对象、并发编程等内容，以及同其他C类语言相比，Go所具备的全新特性。同时本书还介绍了Go语言在云计算中的应用。



基于MVC的JavaScript Web富应用开发

作者：Alex MacCaw

译者：李晶 张散集

本书教你如何构建先进的富应用程序，书中给出的优秀工具和最佳实践都是很多程序员和工程师在工作中亟须的。



软件架构师的12项修炼

作者：Dave Hendricksen

译者：张菲

本书科学而系统地归纳软件架构师应具备的能力和素养，帮助软件开发人员突破技术的“天花板”，成为一名合格的软件架构师。



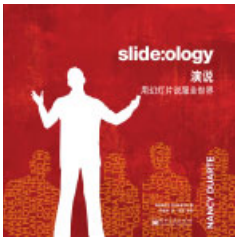
C++代码设计与重用

作者：Martin D.Carroll Margaret A.Ellis

译者：陈伟柱

本书全面展示如何使用C++编写可重用的代码，从而提高程序员的开发效率。

推荐图书



演说：用幻灯片说服全世界

作者： Nancy Duarte

译者： 汪庭祥

出版社：电子工业出版社

这是一本神奇的书。

它颠覆了大多数人，至少是我的线性思维的习惯。我们的脑子里有太多的文字和数字，还有许多模糊的想法，你知道怎样将自己的思想用图形勾勒出来吗？你会把自己的创意画出来吗？目前市面上讲解PPT的书非常多，涉及软件使用、制作技巧、动画设计等各个方面。但别忘了，PPT首先是传递你思想的工具。要让观众随着PPT起舞，重要的前提是吸引观众、打动观众。因此可以说思想性是PPT的灵魂，PPT是承载思想飞扬的两翼，如何让PPT来展现你的思考结晶，正是本书阐述的核心问题。

PPT作为演说中重要的载体和工具，越来越多地受到人们的重视：产品推介、年终述职、岗位应聘、政策说明等，可以这样说，只要有演说的地方，就一定会有投影仪和PPT。你可以憎恨它，却无法抗拒它。往往有人认为，形式不重要，最重要的是内容。话虽如此，但不穿衣服（赤裸裸）的“真理”有谁能看得下去呢？另外，如果没有好的形式与载体，观众收到的只是一堆没点燃的木头或者烧过头的焦炭，而不是你思想的火花，这样的信息传递能算成功吗？因此，当你读完这本书后再看到那些苍白、罗唆、文字堆砌式的PPT，听到那些直白的读稿时，除了感觉时间被糟蹋外，最想做的事情一定是无奈地苦笑不语。

从历史到现实，创意并非国人思维的短板，我们最缺乏的往往是表达的手段和方式。苹果产品中的一些创意，很多国人也有想到过，但没有人考虑过如何以及通过何种方式去实现。如果灵感和创新只停留在脑子里，那就只能永远待在那里。没有人能知道你脑子里有那么多新鲜的想法和改变世界的创意，走在街上，你只是泯然众生中的一个。

如何将所思所想转化为听众理解的内容？这真是一个big problem，但也是证明你是天才的唯一方式。这本书并不会激发你创新的思维或者引导你成为改变世界的乔布斯第二，它的重点在于帮助你把想要表达的东西简单清晰、富有表现力地呈现出来。也许有人看了以后觉得一无所获，因为他脑子里没什么东西需要用这种方式来表达；也许有人看了大有启迪，因为他满腹经纶就缺少新颖的指导和表达的技巧。

本书是西方人所写，在思维习惯和表达方式对国人的启发体现在两个方面：一是思维的逻辑性与严密。从调研、准备到设计、演说等各个阶段均有详述，相对于东方思考注重的宏大，整体而言，在每个阶段的细微之处都能体现西方思维中所特有的精细、严谨与规范。二是细节的技巧。包括每个页面的要素组合、光影的排列、动画的舞动等各个设计细节，都能让读者深受启迪。

通过阅读本书，你能深刻地感受到思考之美、简洁之美以及设计之美。当你读完本书，PPT的制作与设计也许并不再像以往那样枯燥乏味，让你“白头搔更短”。PPT是你的正中靶心的投枪，而不是压垮你的行囊，用PPT来展现你的思考的力量，也许下一个苹果就是你画出来的。（**点评人：刘海泌**）



编写可读代码的艺术
作者： Boswell, D.　Foucher, T.
译者： 尹哲 郑秀雯
出版社： 机械工业出版社

在我们刚开始学习编程时，老师会教我们使用正确的语法、合理的数据结构、高效的算法。而这里的“正确”、“合理”及“高效”，基本都是从机器的角度来说的。但从一般人的角度来看，特别是对非专业人员来说，很多编程语言的语法以及大多数的数据结构和算法都是非常难以理解的。我们可以理直气壮地说“非程序员不需要理解这些东西”，可是我们有没有认真想过，我们编写的代码对于其他程序员来说真的很容易理解吗？

大多数的计算机教育基本只关心代码对于机器的友好性，而不关注代码对于人的友好性。如果代码写完就扔给机器运行，再也没人去阅读和修改，那就没什么问题。可现实情况是，代码会被他人阅读和修改成百上千次。这时，难以理解的代码会浪费程序员大量的时间。在我目前工作的项目中，存在着数百万行的遗留代码，这些代码中的绝大部分都非常难以阅读：变量不停地变换着名字、if语句嵌套个五六层、一个数百行的函数有着数不清的职责……当需要做改动时，程序员不得不花大量的精力去试图理解这些晦涩的代码，而且就算这样也会经常不小心踩到某个陷阱，这种情况非常影响程序员的士气。

因此，代码应该是易于理解的，这正是《编写可读代码的艺术》一书的主旨。使代码变得易于理解的方法多种多样，可以使用DSL（领域特定语言），可以合理使用设计模式，可以使用TDD（测试驱动开发）。相比之下，本书描述的方法看起来最为简单，简单到方法的命名、注释的编写、if/else语句的组织……不过这些看似简单的方法其实很不简单，因为这些细微的地方太难觉察了。

本书的一大特色就是用大量短小的代码说明问题而非长篇理论阐述，每个话题都相应地配有未考虑可读性的代码和考虑可读性的代码，非常便于对比和体会。你可能会认为即便考虑过可读性之后，代码看起来也没有发生太大的变化，无非就是一个变量的名字、一条注释的语言组织、几行语句的结构顺序，不足为道。但我认为这些细微的变化恰恰就是对“艺术”二字最好的诠释。精益求精地编写每一行代码，让他人在阅读时感到心情愉悦，积跬步以致千里，不正是每个程序员应该追求的职业态度吗？

本书在第14章有一个可能会有争议的地方，作者为了使单元测试用例的代码能够尽可能简洁地表达意图，而创建了几个包含很多逻辑的测试脚手架方法。也许有些人会觉得这样的牺牲是值得的，但有些人则觉得这是一种过度设计。这就仁者见仁，智者见智了。

本书每章都包含了许多有趣的漫画，所谓一图胜千言，这些漫画能让你在一笑之余更深刻地理解作者想要表达的意思。本书的封面也非常漂亮，对得起“艺术”二字。如果你希望写出可读性更好的代码，本书值得细细品味。本书在国内有翻译和影印两个版本，大家可以根据自己的喜好挑选最适合自己的版本进行阅读。（**点评人：许晓斌**）

全球排行榜

Amazon

01 HTML and CSS: Design and Build Websites
02 Cracking the Coding Interview: 150 Programming Questions and Solutions
03 Content is Currency: Developing Powerful Content for Web and Mobile
04 The Web Designer's Idea Book
05 iOS Programming: The Big Nerd Ranch Guide, Third Edition
06 Turing's Cathedral: The Origins of the Digital Universe
07 JavaScript & jQuery: The Missing Manual
08 Coders at Work: Reflections on the Craft of Programming
09 Objective-C Programming: The Big Nerd Ranch Guide
10 Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition

天珑书局（中国台湾）

01 《超强图解》前进 Android Market! Google Android SDK 实战演练，2/e
02 JavaScript设计模式
03 JavaScript大全，6/e
04 iOS 创意程序设计家：iPhone + iPad 跨平台通用，3/e
05 App 程序设计入门：iPhone、iPad，2/e
06 iOS SDK开发范例大全
07 HTML5 CSS精致范例辞典
08 Eclipse完全攻略：从基础JAVA到PDE外挂开发
09 深入浅出jQuery
10 精通Objective-C 程序设计，4/e

第二书店

01 数学之美
02 推荐系统实践
03 Java编程思想(第4版)
04 白帽子讲Web安全
05 PHP和MySQL Web开发(原书第4版)
06 神一样的产品经理:基于移动与互联网产品实践
07 Java核心技术,卷1(原书第8版)
08 算法导论（原书第2版）
09 七周七语言：理解多种编程范型
10 性能测试进阶指南——LoadRunner11实战

Mac OS X 背后的故事（十一）

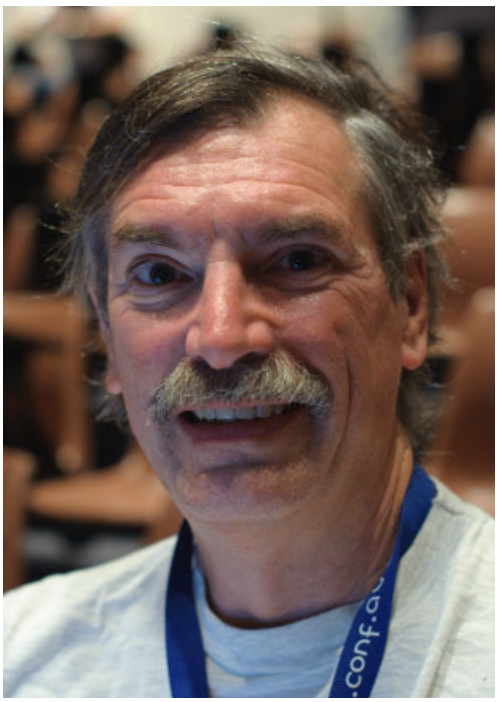
Mac OS X文件系统的来龙去脉（上）

文 / 王越

HFS+和UFS文件系统同时被引入早期的Mac OS X，随着若干年的发展，HFS+提供的功能已超越UFS，使其在Mac OS X 10.5之后成为成为唯一正式的Mac OS X系统，但因为其背负许多的历史包袱，为考虑兼容性，这些陈旧的设计并不能被推翻重来，所以苹果开始秘密研发下一代的文件系统。

UFS：经典的Unix文件系统

在Unix系统刚诞生的远古时期，文件系统被简单地称为FS。FS只包括启动块、超级块（处于硬盘分区开头用来保存文件系统信息）、inodes（索引节点）及数据。FS文件系统在Unix系统刚诞生时还能满足新老客户的需求，但随着科学技术的进步，FS已不能符合现代文件系统的需求，且会导致抖动等一系列问题。当时还是加州大学伯克利分校研究生，后成为著名BSD开发者Marshall Kirk McKusick在BSD 4.1b上承接传统的FS文件系统实现了FFS（Fast File System），妥善地解决了这一难题，把先前整块的磁盘文件系统分为小块，每块包含自己的索引节点和数据，因而增加了文件的局部性，减少了寻道时间。由于Marshall Kirk McKusick的FFS文件系统很好很强大，所以立即被各大Unix系统所使用。SunOS/Solaris、System V Release 4、HP-UX及Tru64 UNIX都使用它，也成为当今各BSD分支（FreeBSD、OpenBSD、NetBSD及DragonFlyBSD）的标准文件系统。每个不同的系统，无论开源与否，又会在FFS文件系统上增加各种扩展，这些扩展往往不互相兼容，但神奇的是，大家又都使用和原版同样的块大小和数据块宽度。因此在很大程度上，这些山寨版FFS文件系统又相互兼容，至少在一个操作系统上能对另一操作系统的文件系统执行只读操作。因此，FFS事实上已经成为Unix系统的标准文件系统，故它有了一个更广泛的称谓——UFS（Unix File System，即Unix文件系统）。



著名BSD开发者Marshall Kirk McKusick

UFS在后来的若干年又取得了长足的发展。Sun公司在Solaris 7系统中，给UFS提供了简单的日志功能。日志文件系统指在档案系统发生变化时，先把相关的信息写入一个被称为日志的区域，然后再把变化写入主文件系统的文件系统。在文件系统发生故障（如内核崩溃或突然停电）时，日志文件系统更容易保持一致性，并且可以较快恢复。Marshall Kirk McKusick又实现了BSD一度引以为豪的Soft Update功能，来保证计算机掉电或系统崩溃时，通过使元数据按依赖顺序更新来确保磁盘上总的文件系统保持一致的实

现机制。Soft Update的目标和日志类似，但实现代价比日志轻量许多。不过这项功能有所代价，主要是需要引入一个后台FSCK检查。2009年，Jeff Roberson正式发表了对UFS的一项改进，为Soft Update加入了日志功能，并消除了对FSCK的依赖，这项改进最终集成进了FreeBSD 9中。TrustedBSD项目又为BSD分支的文件系统设计了ACL访问控制表功能（Access Control Lists）。先前，Unix文件系统的访问控制是非常简单的，其权限管理分为三个不同的类别：用户、同组用户以及其他用户，对每个类别，Unix文件系统提供读、写、执行三种权限的管理。这样的许可管理过于粗糙，无法指定某一用户访问的权限，也无法指定更为细致的权限内容（例如准许对一文件实行删除操作）。为解决这个问题，访问控制表被增加到文件系统中，使用以存取控制矩阵为基础的存取控制方法。存取控制串列描述每一个文件对象各自的存取控制，并记录可对此物件进行存取的所有主体对对象的权限。总之，UFS与时俱进，不断增加新的功能。

HFS+：更现代的HFS

作为Mac OS X的老祖宗NeXTSTEP，因为基于BSD，所以自然也使用UFS。而老版的Mac OS则使用一个叫做HFS的文件系统。HFS是一个比较古老且不思进取的文件系统，因此，在20世纪90年代末已不能满足当时的需要。在《Mac OS X背后的故事（一）》中我们提到，为了实现Mac OS的现代化，Copland项目被提出。Copland项目的子项目Sequoia旨在HFS的基础上，加入现代文件系统所必需的新功能，如大文件支持、Unicode文件名支持、长文件名支持、32位文件映射表支持等。Sequoia项目即成为后来熟知的HFS+，由Don Brady领导，这个团队先花了6个月时间把HFS项目原本的Mac使用的68K处理器汇编码改写成C代码，然后逐渐加入新功能。

后来由于Copland被力挽狂澜的Ellen Hancock给废了，所以一些有用的更新，如HFS+即被集成到Mac OS 8.1中。在Mac OS X诞生初期，HFS+和UFS文件系统同时被引入早期的Mac OS X中。不过由于HFS+根植Mac OS，缺乏Unix文件系统所必需的功能，如符号链接、硬链接及其他各种POSIX兼容性，所以HFS+开发组又花了一些工夫在不影响和Mac OS兼容性的情况下引入了这些功能。由于HFS+是对HFS的扩展，故HFS+支持Mac OS至Mac OS X的平滑过渡，所以Mac OS X一直默认使用HFS+。但当时的UFS提供比HFS+更先进的功能，因此Mac OS X 10.0至10.4，也都支持把系统安装在UFS系统上。

Mac OS X 10.0发布后，苹果不遗余力地对HFS+进行大规模的扩展和维护，增加了很多UFS独有的功能。这些新功能使得文件系统更加安全稳定可靠。例如Mac OS X 10.2.2中，HFS+支持日志。日志功能在Mac OS X 10.2服务器版中可以简单地设定，但在普通桌面版中需要使用命令行进行操作。在Mac OS X 10.3中，带日志功能的HFS+（被称为HFSJ，即HFS+ volume with journal）成为默认设置。Mac OS X 10.3亦增加名件名、目录名区分大小写及Unicode 3.2的支持。Mac OS X 10.4中，HFS+更是增加了ACL访问控制表功能，提供更复杂的对传统Unix文件系统权限的扩展。

文件系统除了让用户供稳定地存放文件这一目标以外，还是各项操作系统功能的基础。Mac OS X每个大发行版都要增加数百项新功能，许多新功能严重依赖于文件系统的实现。Mac OS X 10.3提供了FileVault来加密用户文件，因此用户主目录被保存在一个HFS+文件系统加密镜像中。Mac OS X 10.4提供了系统内置的Spotlight桌面搜寻搜索功能，能让用户对整个磁盘系统进行快速搜寻、随打即显。这项功能要求文件系统提供任意长度文件元数据（metadata）的支持。Mac OS X 10.4转向了对Intel处理器的支持，因此苹果发布了一个测试版本的BootCamp来让用户安装Mac OS X、Windows双系统，并在Mac OS X 10.5正式集成进系统。哪怕在Mac OS X系统运行，BootCamp也可以在时调整系统主分区的大小，来空出磁盘空间给Windows，因此，HFS+又需要支持动态分区大小调整。在Mac OS X 10.5中集成了Time Machine，它是苹果公司所推出备份的工具程序，于2006年8月7日在苹果计算机全球研发者大会（WWDC）中首次公开，成为当天观众

欢呼声最高的功能。Time Machine对于修改过的文件会在备份盘上保存一个新拷贝，而对于不变的内容，仅在备份盘上存一个指向先前文件的硬链接。因此每一次快照只保存改动的文件，而别的文件只保存占用空间很少的硬链接。但Unix一般只支持文件的硬链接而不支持目录的硬链接。因此HFS+在这点上走得比Unix文件系统更远，提供了对于目录的硬链接支持。在Mac OS X 10.6中，HFS+甚至支持文件系统压缩，使得安装后占用比Mac OS X 10.5少得多的空间。Mac OS X 10.7提出了FileVault2，能加密整个磁盘而不是一个用户目录。这些功能我们在为读者介绍每个发行版时亦会提到，但总之读者看到，HFS+的功能随着Mac OS X的商业需求不断被扩展。“我在做了这么多工作后回想才发现，我们为HFS+增加了那么多新功能，”苹果前文件系统开发者Don Brady如是说。

由于HFS+经过后来若干年的发展，提供的功能已不逊于UFS，甚至更多更好，故至Mac OS X 10.5砍掉了安装至UFS的支持。HFS+成为唯一正式的Mac OS X系统。

HFS+并不完美

HFS+自发布以来，几乎每个发行版都有令人欣喜的改动。它也逐渐成为一个非常完善的文件系统。但HFS+立足于HFS设计，HFS已有27年的历史，HFS+亦有14年历史。这个文件系统有大多的历史包袱，为考虑兼容性，这些陈旧的设计并不能被推翻重来。

HFS+基于B-树实现，当查找B-树中未使用的节点时，HFS+只能每次处理16位，原因是老Mac使用的Motorola的68K芯片原生支持16位的数据操作。但不管是PowerPC还是Intel，寄存器都支持256位宽的寄存器。

HFS+的元数据（metadata）都以大字节序保存，原因是Motorola的68k和后来Mac使用的PowerPC都使用大字节序。但经过Intel迁移后，当今的Mac都使用Intel芯片，而Intel芯片是使用小字节序的。因此每当数据读取或存入时，还要经过小字节序和大字节序的转换。远古时期磁盘很慢，计算机处理器的速度也很低，因此进行一次磁盘操作会占用较多的时间，HFS+的时间分辨率为一秒，但当今的磁盘、处理器处理一次文件系统操作的时间远小于一秒，因此所有主流磁盘文件系统的时间分辨率都是一至数百纳秒级别的。

HFS+的元数据有全局锁，同一时间只有一个进程可以访问更新文件系统。在单核处理器连手机平板都较少见到的当今，这种设计显得很幼稚。

HFS+亦没有稀疏文件的支持。例如我们在SQL中建立了一个数据库，SQL分配了10GB的文件给这个数据库，并且在文件头和文件尾写上一些字节的数据。而由于我们还没有给这个数据库添加新的数据，所以这10GB的文件除了头尾外其他字节都为0。现代的文件系统基本都支持稀疏文件，也就是说，当处理这个数据库操作时，事实上往磁盘写入的数据只有那文件头和文件尾的若干字节。而HFS+则需要把那些0也写上，因此会完整写入10GB的数据，耗费长得多的时间。

此外，HFS+不具备元数据校验功能、快照功能、写入时复制功能、就地执行功能逻辑卷管理功能等很多现代磁盘系统所具备的功能，也不能动态调整文件块大小。这些功能的加入并不容易。

其中最要命的是，HFS+不像一些先进的文件系统，支持写入时复制事务模型，也没有快照和克隆。这使得用户数据时时处于风险之中。例如由于因为断电、内核崩溃等原因，文件系统上写到一半的数据，小则导致个别文件损坏，大则导致整个文件系统崩溃。在生产领域，这样不可靠的文件系统，很有可能带来致命的灾难。

正是由于上述这些原因，连我们介绍过的短视的Linus Torvalds都认为HFS+是个垃圾文件系统。苹果自然受不了这种侮辱，因此，干掉HFS+势在必行。用什么取代HFS+呢？苹果开始秘密研发下一代的文件系统。



王越

美国宾夕法尼亚大学计算机系研究生，中国著名TeX开发者，非著名OpenFOAM开发者。

责任编辑：陈博（chenbo@csdn.net）

MSUP三步曲：MPD之因团队施教

聚焦软件研发中心的快速成长

“MPD”是Make（改变）、Professional（专业）、Discovery（探索）的缩写，2010年由麦思博（MSUP）有限公司创办。它是一个课堂，是一个大会，是一个互动的论坛，在这里可以分享到苹果、微软、谷歌、亚马逊、惠普、IBM等跨国研发团队一线带头人的成功、喜悦、失败和沮丧，你无须跑到美国总部或商学院去，它就在你伸手可及的地方，你相信吗？这就是MPD。

MSUP总经理刘付强用5年的时间思考软件行业发展趋势、思考软件行业中的企业组织和人的关系以及培训与软件行业的关系，经历了一个痛苦的孕育、成长和涅槃的过程。

思考

2003年无论是对培训行业来说还是对软件行业来说，都是暗流涌动的一年。刘付强这样描述当时的情况：我感觉当年的IT培训公司相对于客户来说像是个拙劣的厨师，不管客人想吃什么样的菜，我们都会端上相同的菜肴，仅有的变化是可能菜的名字不一样，当然结局也是可想而知的。最终大家只能拼价格，陷于典型的“低端竞争”，最终导致整个IT培训环境的恶化。

打破现有的培训模式，需要对软件行业的载体行业、企业和人有深刻的了解。刘付强和他的团队决定开拓出一片属于自己的蓝海，“遵从孔老夫子的教诲，我们设计了三个菜系：因人施教、因团队施教、因企业施教，之所以提出这三个施教，就是我们认为IT培训的发展应当伴随着信息产业的前行步伐。”

孕育

2003年是MSUP的元年，当年刘付强认为软件行业的年增长率将达到20%以上，而MSUP将自身的师资、销售人员储备年增涨率设定为25%。八年间，软件产业规模扩大了十几余倍，年均增长38%，略显保守但是准确的研判使MSUP看到了即将到来的大潮，那么如何能够站在潮头呢？

2003年同样也是软件业开发模式发生变化的起点，日趋庞大的软件项目和复杂需求需要更多人以标准化的软件工程方法去实现。这个过程科学而严谨，软件开发的每个步骤开始细分。那是一个转折多变的年代，但是团队开发尚未形成气候，同时结合着中国企业文化的特点、“技术大拿制”在国内软件企业中颇为盛行的现状，MSUP推出了其“三大菜系”的第一道——“因人施教”，同时全力丰富课程资源，不断优化课程内容，由此凝聚了国内外大批优秀的教师资源和大浪淘沙下的大量优秀课程，使MSUP掌控的核心培训内容达到了一个空前丰富的程度，为MPD的第二步打下了良好的基础。

成长

作为这个行业最基础生产力的构成，开发者不应该只有剩余价值，还应该有梦想。我们缺少的只是能将梦想放大而又不会与现实背道而驰的机会。程序员虽然再不会成为从前的个人英雄，但是也不会沦为软件工厂中的代码机器，而是在团队背景下的集体团队行动，角色分工和技术技能同样重要。

“个人英雄或技术大拿式的开发将向团队开发转变，团队式开发模式将成为今后软件行业的趋势。”刘付强和他的团队如此判断2005年后开发模式的变化“对开发模式预判促使我们在‘因人施教’的基础上推出了第二道菜‘因团队施教’，也就是现在展现在大家面前的MPD”。

MPD在酝酿了5年之后于2010年3月20日诞生了。它开创性地将CMMI类重量级团队模式、IBM RUP模式、MSF模式、敏捷团队模式进行了整理、综合、分解和再整合，根据我国软件企业的现状分解出了产品经理、团队经理、架构师、开发经理、测试经理五种角色，将所有的技能融入这五种角色。《程序员》杂志总编、图灵出版创始人刘江曾经评价：“这可能是技术领域最好的管理培训。”

明天

9年前的“因人施教”还是7年前“因团队施教”虽然使MSUP取得了一定的成功，但是也使MSUP的团队认识到，个体或者一个团队的成功并不能保证一家软件企业的成功，而中国的软件企业正是MSUP希望能够帮助的目标客户群体，因此，站在“因团队施教”的肩膀上，MSUP将再用5年的时间去打造自己的满汉全席，以实现“因公司施教”的目标。



Stephen Wolfram

智能搜索引擎Wolfram|Alpha创造者

名人堂

“新”科学家

Stephen Wolfram

文 / 卢鹞翔

2011年10月的iPhone 4S发布会上，神奇的智能助手Siri就此横空出世，名闻天下。而大部分人并不知道，Siri许多答案都来自Wolfram|Alpha这个智能搜索引擎。Stephen Wolfram是它的创造者，一位具有传奇色彩的科学家。

Wolfram1959年生于伦敦，他几乎获得了所有少年天才能得到的成就——13岁进入伊顿公学，15岁发表首篇关于粒子物理的学术论文，在牛津大学学习一年之后，进入美国Argonne国家实验室工作。1978年，19岁的Wolfram受诺贝尔奖得主Murray Gell-Mann之邀到加州理工学院从事物理研究，一年内获得了理论物理博士学位，继而被学校聘用，从1980年开始与费曼共事。1981年，为了表彰他在物理和计算机领域的早期创新工作，Wolfram获得了麦克阿瑟奖，是当年该奖项最年轻的获得者。

Wolfram从14岁时开始接触计算机，“我一直认为，做研究一定要用最好的工具。70年代中期，计算机在英国还很稀罕。我第一次接触，用的还是纸带，但我很快就成为一个厉害的程序员。”Wolfram回忆说。当时恰逢科学计算兴起，Wolfram很快成为这一领域的先锋，他1979年开始构建SMP（最早的现代计算机代数系统），并在1981年发布了商业版本。

20世纪80年代中期，个人电脑的计算能力不断增强，Wolfram使用不同软件，完成不同领域的研究，他设想开发一个通用的计算系统，让所有的研究工作只在这个系统中完成。“我还想要更好的工具，我明白，唯一的方法就是自己将它们造出来。”——这个系统就是Mathematica，名字来源于当时刚离开苹果的史蒂夫•乔布斯的建议（完全符合乔教主的取名原则：在通用名的基础上略做修改），并与NeXT计算机一同在1988年亮相。谈到对Mathematica的期望时，Wolfram这样说：“如同伽利略400年前用望远镜观察宇宙，我希望用Mathematica观察可计算的宇宙。”经过20多年的开发，如今它是使用最广泛的数学软件之一。在此之前的一年，Wolfram还建立了以他自己名字命名的Wolfram Research研究中心，除了进行Mathematica的后续开发，Wolfram|Alpha也是Wolfram Research的成果。

2009年发布的Wolfram|Alpha最初被外界看做Google杀手，但两者的定位显然有区别很大，Wolfram|Alpha是问答引擎，它针对使用者提出的问题，直接给出答案，并显示相关信息。在谈到与其定位接近的IBM Watson的不同之处时，Wolfram表示，Watson的原理与搜索引擎接近，都是通过统计学匹配和过滤答案，而Wolfram|Alpha并非基于传统的统计学原理，它有自己的知识计算引擎和相应领域的数据库，答案是“算”出来的，而不是“找”到的。基于统计的搜索引擎只能回答已有答案的问题，而Wolfram|Alpha能解决未知问题。

不过Wolfram Alpha也有它的不足，它对自然语言理解能力有限，有时只能理解一句完整

输入中的几个关键词，另外也只有在具备相应领域的知识数据库前提下，才能解答这个领域的问题。不过它的理解能力已越来越强，2009年发布伊始，它只能理解60%的输入，而如今这个数字已超过90%。或许未来的某一天，我们可以看到Wolfram|Alpha拥有足够能力解决Stack Overflow上的问题。

可计算文档格式——CDF是Wolfram Research的另一项创造。今天的在线文档与传统书本大致相似：平白、缺乏互动元素。CDF的核心理念是创作具有互动性，让读者可以驾驭内容，实时生成结果。

Wolfram还是电子书出版公司Touch Press的联合创始人，这家公司的目标是利用最新的技术手段，重新定义书籍，改变人们的阅读方式。iOS上的热门应用《The Elements》、《Solar System》都是Touch Press的作品。此外Wolfram Research还是CBS流行美剧《Numb3rs》的数学顾问。

1981年，Wolfram的研究兴趣从理论物理转向了细胞自动机，基于这一理论，他在20世纪80年代取得了一系列成果。例如创造了一套新的随机生成系统，以及新的计算流体动力学模型，这两项创造在今天都得到了广泛应用。在对细胞自动机的研究过程中，他还逐渐意识到其中可能蕴涵着解决自然界中复杂性起源问题的答案。

1991-2002年的4000多个夜晚里，Wolfram敲击了1亿次键盘，移动了100多英里鼠标，做了上万页的笔记，编写了近百万行的Mathematica程序，产生的研究结果占据了10G硬盘空间。（从1989年，Wolfram就开始记录所有邮件、作息、敲键、电话、运动等个人信息，他认为这些信息不但为我们感受生活增加了一个新维度，还能用于分析过去，预测未来）。终于完成了一本将近1200页、5.6磅的大部头——《A New Kind of Science》（NKS）。

当作品完成后，Wolfram曾邀请乔布斯在封底写一段推荐语，乔布斯说“牛顿著作的封底没有推荐语，为什么你需要呢？”所以NKS最终以简洁的图片作为封底——艾萨克森的乔布斯传记也是相同的风格。

“三个世纪以前，人们用数学描述自然界的思想，使科学发生了极大转变。在这本书中，我将尝试发起另一次转变，建立一种新科学，它基于更基本的法则，并且可以通过简单的计算机程序将其具体化。”Wolfram在开篇中这样写道。如同爱因斯坦的统一场论，Wolfram希望自然界中一切复杂性问题都能通过这套理论解决。他要用细胞自动机这种简单的电脑程序取代数学方程，这位曾经的物理学家，似乎要用计算机革物理学的命。

然而这本书在出版之后遭到了很多非议，一方面Wolfram在自己并不擅长的领域发表了惊人结论，让其他领域的专家难以接受。例如只根据简单的例证，就得出了生物差异与自然选择基本无关的结论。另一方面，书中关于细胞自动机的很多思想是受前人启发，但字里行间不自觉夸大了自己的工作。但这并不影响它的热销，2002年5月出版后的几周，热度超过了当时许多热门小说，一度高居Amazon畅销书榜首。

今年恰逢NKS出版10周年，Wolfram正在着手为它增加新的一章，在尚未出版的手稿中他这样写道。

- 新科学的阶段：吸收（尽量理解书中所写，初步理解需要两年，进一步需要5年）；初步延伸（2～3年，10～15年后结束）；建立主要新方向（15～30年）；小规模早期技术应用（4～10年）；大规模技术应用（10～25年）；成为常识（4～10年）；成为基础科学教育的一部分（15～20年）。

- 原则：寻找最简洁的案例，解决最显而易见的问题；不要满足于技术解释，要探寻事件的根本原因；不要被前人的工作束缚，但要尽可能全面理解这些工作；尽可能清晰地解释你的工作，少依赖底层架构。

这或许可以作为Wolfram对人们质疑的回应，也可以作为他科学生涯的注解。



西乔
设计师，项目经理。
2006年起携创业团队从事Web技术外包开发及产品咨询顾问。

如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，
请给西乔发邮件：arthur369@gmail.com

改编自It Takes 6 Days to Change 1 Line of Code
http://edweissman.com/it-takes-6-days-to-change-1-line-of-code